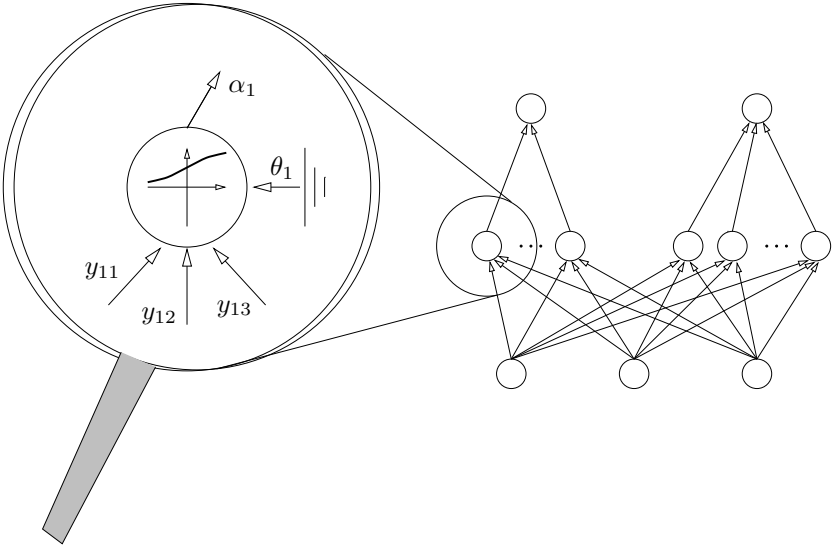
Zur Theorie künstlicher neuronaler Netze

Book

Stefan M. Rüger

Zur Theorie künstlicher neuronaler Netze



Unveränderter Nachdruck, zuerst erschienen 1997 beim Verlag Harri Deutsch, Thun, Frankfurt am Main (© 1997–2000, Reihe Physik, Band 71, ISBN 3-8171-1542-3); zugleich Dissertation D 83, Technische Universität Berlin, 16 Dez 1996

Für meine Eltern, Irene und Josef Rüger,
denen ich eine ausgesprochen schöne
Kindheit verdanke.

Abstract

Zur Theorie künstlicher neuronaler Netze wird aus vier Gebieten beigetragen: der Informatik mit einem neuen Lernverfahren (stabile Parameteradaption), der Mathematik mit der Analyse der Struktur des Gewichtungsraums, der Statistik mit einem neuen Schätzer für die Güte von Netzen (Clustered bootstrap) und der Physik mit effizienten Lern- und Schließalgorithmen für dezimierbare Boltzmann-Maschinen.

Es werden Abbildungsnetze definiert, deren Kettenregel abgeleitet und in mehrere berechtigte algorithmische Varianten gefaßt, Backpropagation-Netze definiert, der Backpropagation-Algorithmus in einer möglichst allgemeinen Fassung dargestellt und demonstriert, wie dieser Rahmen auch auf rekurrente Netze angewendet werden kann.

Die Grenzen der Methode des Gradientenabstiegs werden aufgezeigt und bekannte alternative Verfahren kritisch dargestellt. Ausgehend davon wird unter den Gesichtspunkten Effizienz und Stabilität eine Klasse neuer miteinander verwandter Optimierungsalgorithmen entwickelt, deren theoretische Leistungsfähigkeit von einem Beweis der Konvergenz erster Ordnung abgesichert wird. Es ist möglich, Zweite-Ordnung-Information in das neue Verfahren einfließen zu lassen. Empirische Vergleiche untermauern dessen Effizienz. Die Grenzen von Optimierungsverfahren werden diskutiert.

Danach wird Lernen in neuronalen Netzen als statistisches Schätzproblem aufgefaßt. Die Güte der Schätzung kann mit bekannten statistischen Verfahren berechnet werden. Es wird nachgewiesen, daß durch Unzulänglichkeiten neuronalen Lernens die Angaben zur Güte nicht robust oder zu ungenau sind.

Das Bestreben, diese Unzulänglichkeiten herauszufiltern, führt auf eine neue theoretische Sichtweise des Gewichtungsraums. Er muß in natürlicher Weise als Mannigfaltigkeit verstanden werden. Es zeigt sich, daß die Berechnung der kanonischen Metrik im Gewichtungsraum NP-hart ist. Zugleich wird nachgewiesen, daß eine effiziente Approximation der Metrik möglich ist. Damit ist es möglich, Lernergebnisse im Gewichtungsraum zu clustern und zu visualisieren. Als eine weitere Anwendung dieser Theorie wird ein robustes Verfahren der Modellauswahl vorgestellt und an einem Beispiel vorgeführt. Schließlich kann auch das im vorigen Absatz gestellte Problem durch ein neues Verfahren gelöst werden.

Die physikalisch motivierte Boltzmann-Maschine wird dargestellt, und es wird argumentiert, warum hier das Schließen NP-hart ist. Dies motiviert eine Beschränkung auf die genügend interessante Klasse der dezimierbaren Boltzmann-Maschinen. Eine neue Dezimierungsregel wird eingeführt und gezeigt, daß es keine weiteren gibt. Dezimierbare Boltzmann-Maschinen werden mit Mitteln der Wahrscheinlichkeitstheorie studiert und effiziente Lernalgorithmen vorgeschlagen. Die Gewichtungsraumstruktur kann auch hier erfolgreich ausgenutzt werden, was eine Anwendung demonstriert.

Vorwort

There is a theory which states that if ever anyone discovers exactly what the Universe is for and why it is here, it will instantly disappear and be replaced by something even more bizarre and inexplicable.

There is another theory which states that this has already happened.

— D. N. Adams, „The Restaurant at the End of the Universe“

Jedes wissenschaftliche Forschungsgebiet braucht für seine Glaubwürdigkeit eine solide theoretische Grundlage. Das Gebiet der neuronalen Netze scheint gleich mehrere unterschiedlich motivierte Fundamente zu besitzen. Es ist hochgradig interdisziplinär und wird von Biologie, Informatik, Ingenieurwissenschaften, Mathematik und Physik bewirtschaftet.

Mein Interesse an neuronalen Systemen ist nicht primär biologisch motiviert. Aber Erkenntnisse der Neurobiologie können zur Lösung von Problemen in Natur- und Ingenieurwissenschaften beitragen. Die dabei zur Anwendung kommenden *neuronal inspirierten* Netze mögen neurobiologische Relevanz besitzen oder auch nicht. In diesem Sinn ist das Gebiet der *künstlichen neuronalen Netze* umfassender als das eigentliche Gebiet der (biologischen) neuronalen Netze. Ja man könnte sogar argumentieren, anstatt von künstlichen neuronalen Netzen lieber von *konnektionistischen Systemen* oder vom *Parallel distributed processing* zu sprechen. Beiträge zur Theorie künstlicher neuronaler Netze stammen aus vielerlei Gebieten:

– aus der Informatik, etwa durch das Studium massiv paralleler Algorithmen, der Komplexitätstheorie oder des Designs von Hardware,
– aus der Mathematik, z. B. aus der konvexen Optimierung, der Funktionalanalysis mit Ergebnissen, welche Funktionen approximiert werden können, und besonderes aus der Statistik durch Arbeiten zur Mustererkennung und -klassifikation

– und aus der Physik, in der künstliche neuronale Netze mit Methoden der statistischen Mechanik behandelt werden können.

Der letzte Punkt verdient etwas mehr Erläuterung: Die Thermodynamik, aus der die statistische Mechanik hervorgegangen ist, hat alle physikalischen Revolutionen dieses Jahrhunderts, die unser Verständnis von der Zusammensetzung und der Wechselwirkung elementarer Bausteine der Materie grundlegend verändert haben, schadlos überlebt.♮ Es scheint so, als ob die thermodynamische Beschreibung der Materie sehr wenig mit der inneren Beschaffenheit der Materie und ihrer Wechselwirkung zu tun hat, sondern vielmehr aus physikalischen Systemen etwas Grundlegendes, nicht an die physikalische Realisierung des Systems Gekoppeltes, abstrahiert: ihre Organisationsstruktur oder ihre Informationseigenschaften.

Erst gegen Ende der zwanziger Jahre hat Leo Szilard die lange vorher von Ludwig Boltzmann als Maß für Unordnung gegebene Interpretation der Entropie verschärft und Entropieverringerung mit Informationszuwachs identifiziert. Seither muß Information als feste physikalische Größe♭ betrachtet werden. Unter diesem Blickwinkel erscheint es natürlich, daß die Thermodynamik als empirische Wissenschaft zusammen mit der Theorie der statistischen Mechanik geeignete Werkzeuge zur Untersuchung kooperierender, massiv paralleler, kleiner, informationsverarbeitender Einheiten zur Verfügung stellen kann.

Die letzten beiden Kapitel dieser Arbeit reflektieren diesen Standpunkt. Was nicht heißt, daß zu deren Verständnis ein physikalischer Hintergrund notwendig wäre. Auch erscheint es mir nicht hilfreich, die physikalische Bedeutung der Begriffe wie Energie, Temperatur und Entropie♭♮ einzuführen, denn ihre informationstheoretische Interpretation ist im Kontext dieser Arbeit weitaus bedeutender und einleuchtender. Vorausgesetzt wird je-

♮ Ausnahme: Das dritte Axiom der Thermodynamik wird durch Annahme der Gültigkeit der Quantenmechanik sogar zu einem Theorem.

♭ mit Einheit Joule pro Kelvin: Ein Bit ist $k \log(2) \approx 9{,}57 \cdot 10^{-24}$ J/K, wobei die Boltzmann-Konstante k die Einheiten von Energie und Temperatur ineinander umrechnet. Daß ein Bit sich so darstellt, folgt sofort aus einem schon 1877 von Ludwig Boltzmann aufgestellten Zusammenhang $S = k \log \Phi$ zwischen der Entropie S und dem Phasenraumvolumen Φ.

♭♮ für neugierig Gewordene: ⟦Falk und Ruppel 1976, Callen 1985⟧

doch Mathematik, insbesondere mehrdimensionale Analysis, lineare Algebra und Grundzüge der Wahrscheinlichkeitstheorie und Statistik.

Der Hauptakzent dieser Arbeit liegt in der Theorie. Daher sind praktische Anwendungen, wo sie vorhanden sind, eher als Demonstration der Relevanz des zugrundeliegenden Theorems konzipiert. Ich habe in besonderem Maß Wert darauf gelegt, nicht nur die bestehende Theorie so weiterzuentwickeln, daß sie Einsichten in die Struktur künstlicher neuronaler Netze gibt, sondern auch darauf, daß sich meine theoretischen Ergebnisse praktisch umsetzen lassen. Die Notation ist daher so explizit gewählt, daß eine Implementierung der wichtigsten Resultate direkt erfolgen kann, ja in einigen Fällen ist neben der mathematischen Aussage auch der zugehörige Algorithmus angegeben. Die numerische Korrektheit einer daraus abgeleiteten Implementierung (wie Vorsorge gegen Bereichsüberschreitungen) bleibt jedoch dem Anwender überlassen.

Berlin, im August 1996 *Stefan Rüger*

Notation

Even the town mathematician confessed that he could make nothing of so dark a problem. X, everybody knew, was an unknown quantity; but in this case (as he properly observed), there was an unknown quantity of X.

— E. A. Poe, „X-ing a Paragrab“

Die Anwendung einer Funktion $f\colon A \to B$ auf Elemente x des Definitionsbereichs A wird meist mit $f(x)$, f_x oder f^x notiert. Wo das Funktionssymbol f schon eine andere Bedeutung hat, wird anstelle von f auch gleichwertig $f(\bullet)$, $x \mapsto f(x)$, $f_\bullet$ usw. benutzt. Hierbei ist x als gebundene Variable zu verstehen. Ist $T \subset A$ eine Teilmenge des Definitionsbereichs A, so bedeute $f(T)$ die Menge $\{b \in B \mid b = f(t) \text{ für ein } t \in T\}$.

Tiefindizes deuten oft auf Komponenten eines Vektors hin; dabei können Vektoren mit beliebigen Symbolen indiziert werden. Sei beispielsweise N eine Indexmenge; dann ist $x \in \mathbb{R}^N$ ein Vektor, dessen Komponenten x_a mit Indizes aus $a \in N$ referenziert werden. Die Transponierung eines Vektors x wird mit x^T notiert. Hochindizes bedeuten ansonsten meist Elemente einer Folge: x^i ist dann das i-te Folgeglied. Manchmal ist jedoch auch die i-te Potenz gemeint. f^{-1} ist die Umkehrrelation einer als Relation aufgefaßten Funktion f.

Explizite Produkte $a \cdot b$ und implizite Produkte ab unterscheiden sich nur in ihrer Priorität:

$$2/ab = 2/(ab), \quad \text{aber} \quad 2/a \cdot b = (2/a) \cdot b$$

Manche Symbole haben im Lauf dieser Arbeit mehrere Bedeutungen. Dies liegt meistens daran, daß in verschiedenen Gebieten etablierte Bezeichnungen beibehalten wurden, z. B. steht E sowohl für die Kantenmenge als auch für die Fehlerfunktion. E steht in dieser Arbeit für den Erwartungswert, und der aufmerksame Leser wird bemerkt haben, daß E und E sich leicht

im Schriftsatz unterscheiden. Unabhängig davon wurde darauf geachtet, daß dem Kontext zweifelsfrei die Bedeutung des jeweiligen Symbols zu entnehmen ist. Umgekehrt stehen in Indizes die Symbole o und 0 beide für die Null: w_{oa} sieht im Schriftbild viel besser aus als w_{0a}.

Wichtige Mengen

$\emptyset$	leere Menge
$\mathbb{N}$	Menge $\{1, 2, \ldots\}$ der natürlichen Zahlen
$\mathbb{Z}$	Menge der ganzen Zahlen
$\mathbb{R}$	Menge der reellen Zahlen
$\mathbb{C}$	Menge der komplexen Zahlen
$\mathbb{N}_o$	$\{0\} \cup \mathbb{N}$
$\mathbb{R}^+$	Menge der positiven reellen Zahlen
$\mathbb{R}_o^+$	Menge der nichtnegativen reellen Zahlen
$\mathbb{R}^n$	n-dim. Vektorraum mit Skalarprodukt $xy = \sum_{i=1}^{n} x_i y_i$
$[a, b]$	abgeschlossenes Intervall von a bis b, $a \leq b$
(a, b)	offenes Intervall von a bis b, $a < b$
$\mathbb{I}$	abgeschlossenes Einheitsintervall $[0, 1]$

Wichtige Operationen auf Mengen

$M \times N$	Kartesisches Produkt: Menge der Paare (m, n) mit $m \in M$ und $n \in N$
M^N	Menge aller Abbildungen von N nach M
$\mathbb{R}^N$	wird als Vektorraum oft identifiziert mit $\mathbb{R}^{\lvert N\rvert}$
$\lvert M\rvert$	Mächtigkeit einer Menge M

Im Text definierte Symbole

Im Anschluß an das Literaturverzeichnis findet sich eine Übersicht der im Text definierten und benutzten Symbole. Es ist zum schnellen Nachschlagen für den Leser gedacht. Nicht jedem im Text definierten wichtigen Begriff konnte eine prominente Unterabschnitts-Überschrift gegeben werden; im Namen- und Sachverzeichnis am Ende dieser Arbeit sind jedoch alle relevanten Begriffe gelistet.

Inhaltsverzeichnis

Sometimes I think the surest sign that intelligent life exists elsewhere in the universe is that none of it has tried to contact us.

— Calvin & Hobbes, „Weirdos From Another Planet!“

Abhängigkeit der einzelnen Abschnitte

Ein Mops kam in die Küche und stahl dem Koch ein Ei. Da nahm der Koch ...

— Kinderlied

Die in Abb. 1 gezeigte Abhängigkeit der Abschnitte wurde automatisch von SmrTEX erzeugt. Durchgezogene Pfeile bedeuten „wird benötigt für“ und gestrichelte „verweist auf Bedeutung für“.

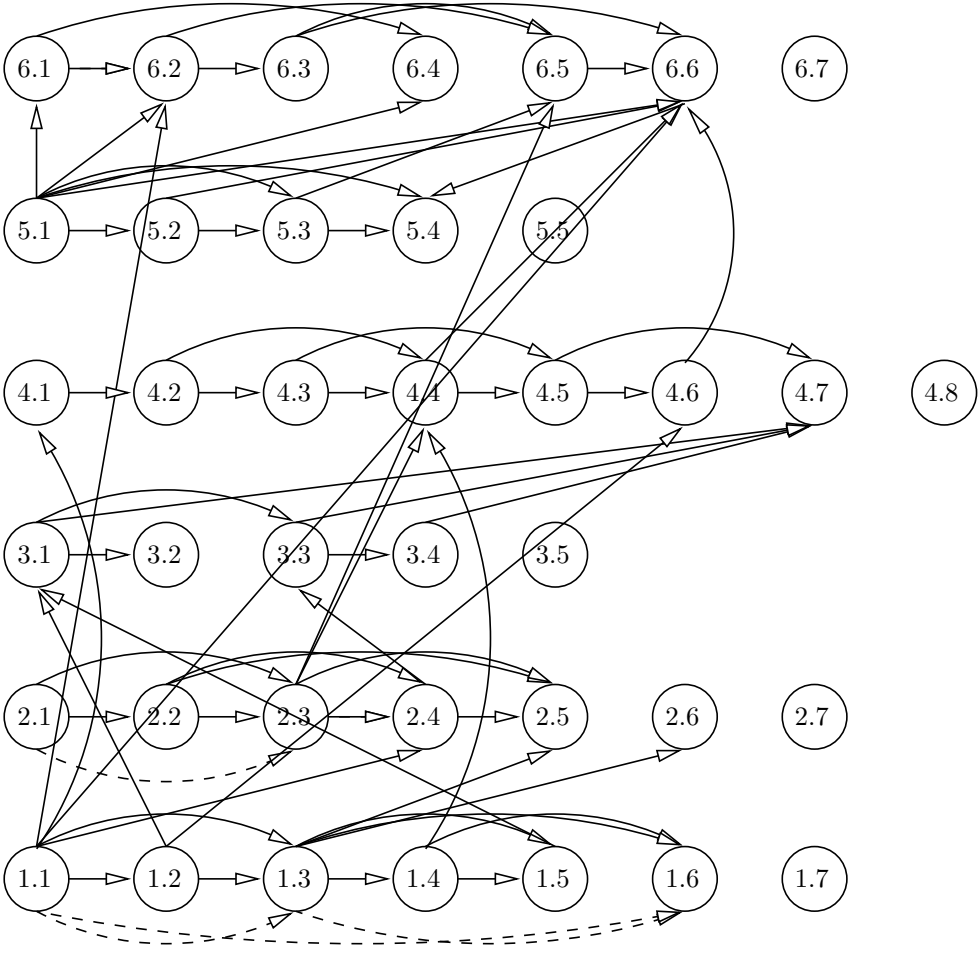


Abb. 1 Abhängigkeit der Abschnitte

Abbildungsverzeichnis

Worte, die geschriebene oder gesprochene Sprache, scheinen in meinem Gedankenapparat keine Rolle zu spielen. Die physischen Gebilde, die als Elemente des Denkens dienen, sind gewisse Zeichen und mehr oder weniger klare Bilder [...]

— A. Einstein nach ⟦Hadamard 1945⟧

Tabellenverzeichnis

Nur das Gründliche ist wahrhaft unterhaltend.

— T. Mann

Einleitung

Wenn das einzige zur Verfügung stehende
Werkzeug ein Hammer ist, neigen Probleme
dazu, so auszusehen wie ein Nagel.

— unbekannte Quelle

Computer sind aus unserer heutigen Welt nicht mehr wegzudenken. Sie sind eine wertvolle Hilfe, etwa beim Lösen wissenschaftlicher und mathematischer Probleme, beim Erstellen und Anwenden von Datenbanken, zur Regelung komplexer Systeme, für die elektronische Kommunikation, für die Textverarbeitung, für die Erzeugung und Bearbeitung von Grafiken usw. Trotzdem gibt es eine Reihe von Problemen, bei denen eine Automatisierung erwünscht ist und die herkömmliche Computer nicht befriedigend bewältigen können: Spracherkennung, semantisches Verstehen von Texten, Segmentierung von Bilddaten, Mustererkennung, Bewegungskontrolle von Robotern, autonomes Führen von Fahrzeugen u. v. m.

Kraß ausgedrückt: Wenn es um das Erkennen und Auffinden von Nahrung geht, erweist sich jeder Kakerlak als deutlich effizienter als ein von Experten programmierter Supercomputer. Woher rührt diese Diskrepanz? Zum einen sicher daher, daß die Möglichkeiten der traditionellen Künstlichen Intelligenz und der Softwareentwicklung noch nicht ganz ausgereizt sind. Zum anderen aber scheint es so, daß Fehlschläge bei dem Versuch, das Anwendungsgebiet von Computern zu vergrößern, durch die heute durchgängig verwendete Rechnerarchitektur bedingt sind: Von-Neumann-Rechner haben eine sequentielle Struktur, und die dafür geschriebenen Programme müssen präzise ausgearbeitet sein. Ein falsch gesetztes Komma kann Schäden in Millionenhöhe verursachen.

Vorbilder aus der Natur zeigen wie Rechnerarchitekturen sein sollten: massiv parallel, fehlertolerant, adaptiv, flexibel, klein und mit niedrigem Energieverbrauch. Künstliche neuronale Netze stellen ein *alternatives Rechner- und Programmierparadigma* dar, das diesen Forderungen noch am nächsten kommt. Sie werden nicht gemäß einem Flußdiagramm Bit für Bit programmiert, sondern lernen anhand von Beispielen, mit ihrer Umgebung zu inter-

agieren. Ein typisches Netz beinhaltet dabei *Parameter* oder *Gewichte*, die einerseits das Netzverhalten bestimmen und andererseits verändert werden können. Das Gelernte wird in Form von adaptierbaren Gewichten verteilt gespeichert. Das Programmieren der Netze beschränkt sich in den meisten Fällen auf die Angabe von *Lernregeln*, die bestimmen, wie die Gewichte bei der Präsentation neuer *Muster* (d. h. Beispiele) geändert werden sollen, um mit der neuen Situation zurechtzukommen. Ziel ist es meistens, eine sog. *Kostenfunktion*, die die anfänglich vom Netz gemachten Fehler bestraft, zu minimieren. In solchen Fällen handelt sich das Lernen in neuronalen[♮] Netzen um ein Optimierungsproblem.

Eine generelle Unterscheidung bei Lernalgorithmen in neuronalen Netzen wird dadurch gegeben, ob dem Netz beim Lernen eine richtige, erwünschte Antwort vorgegeben wird (Supervised learning, Lernen durch einen Lehrer), ob eine Bewertung des Verhaltens gegeben wird (Reinforcement learning, Lernen durch einen Kritiker) oder nichts dergleichen (Unsupervised learning, unüberwachtes Lernen). Im letzteren Fall kann aber sehr wohl argumentiert werden, daß im Prinzip die gewünschte Netzausgabe mit der Netzeingabe identisch ist (sog. autoassoziatives Netz), und daß das Netz in irgendeiner Form eine Datenkomprimierung der Eingaben vornimmt ⟦Sarle 1994, Deco und Obradovic 1996⟧. In der klassischen Statistik sind diese „Lernverfahren“ als Clusteranalyse bekannt. Durch die Beschäftigung mit neuronalen Netzen sind i. w. selbstorganisierende Karten zum bestehenden Bereich der Clusteranalyse hinzugekommen. Sie sind vor allem bei der biologischen Modellierung von neuronalen Systemen von Interesse. In der vorliegenden Arbeit werden zwar bekannte Clusteralgorithmen benutzt und autoassoziative Netze gestreift (bei rekurrenten Netzen und dem Hopfield-Netz als Spezialfall der Boltzmann-Maschine), aber ansonsten nicht tiefer studiert.

Für den vor allem in technischen Anwendungen wichtigen Bereich des Supervised learning sind inzwischen sehr viele dieser Lernregeln vorgeschlagen worden – und man hat erkannt, daß viele bekannte Methoden z. B. der Statistik in das Gebiet fallen. In Ermangelung an geeigneter Hardware

[♮] Der Einfachheit halber und im Einklang mit dem geltenden Jargon wird das Adjektiv künstlich weggelassen, obwohl – wie im Vorwort argumentiert – es zutreffender wäre, das Adjektiv neuronal wegzulassen.

werden üblicherweise neue Lernregeln auf herkömmlichen Computern simuliert, um empirische Erkenntnisse zu gewinnen. (Wenn das einzige zur Verfügung stehende Werkzeug ein sequentiell arbeitender Computer ist, neigt man dazu, Problemlösungen einer sequentiellen Programmierung zu unterwerfen.)

Bei diesem Stand der Forschung ist es umso nützlicher, wenn theoretische Ergebnisse bei der Suche nach geeigneten Lernregeln helfen oder dazu beitragen, bestehende Lernverfahren oder resultierende Anwendungsergebnisse besser beurteilen zu können. Das Anliegen, solche theoretischen Einsichten zu gewinnen, zieht sich wie ein roter Faden durch meine gesamte Arbeit.

Im ersten Kapitel wird der wohl populärste Lernalgorithmus für neuronale Netze, Backpropagation, in einer sehr allgemeinen Form dargestellt. Ein Schlüssel für die von mir gegebene Darstellung war die Isolierung eines primitiven, aber sehr nützlichen Werkzeugs: die Kettenregel in allgemeinen Abbildungsnetzen. Sie wird in dieser Arbeit mehrfach benutzt, nicht nur, um mehrere bestehende Lernalgorithmen in einer einheitlichen Weise zu beschreiben und elegant zu beweisen, nicht nur, um durch Mehrfachanwendung eine effiziente Berechnung der Hesseschen Matrix mit Abbildungsnetzen vorzuschlagen, sondern auch, um neue effiziente Lernalgorithmen für einen völlig anderen Netztyp (die Boltzmann-Maschine) zu definieren.

In Kapitel 2 wird von der – in der Praxis viel zu häufig verwendeten – Methode des Gradientenabstiegs gezeigt, daß sie deutlichste Mängel aufweist: Ein Trade-off-Theorem zeigt, daß sie in der Regel entweder langsamer als nötig konvergiert oder gar nicht. Der Grund dafür ist in fehlender Stabiltät zu suchen. Nach einer kritischen Analyse und Darstellung bekannter alternativer Optimierungsverfahren wird eine Klasse neuer miteinander verwandter Lernverfahren, die stabilen Parameteradaptionen, entwickelt. Ein Konvergenzbeweis zeigt, daß für jeden Algorithmus dieser Klasse (der erst durch die Wahl der Optimierungsrichtungen festgelegt wird) asymptotisch ein Optimum mit mindestens linearem Konvergenzverhalten gefunden wird. Jedoch ist ein superlineares Verhalten durch geschickte Wahl der Optimierungsrichtungen (hier durch konjugierte Gradienten gegeben) möglich, was anhand von empirischen Untersuchungen veranschaulicht wird. Die in diesem Kapitel vorgestellten Versuchsergeb-

nisse weisen die Effizienz der Lernverfahren für praktisch relevante Fälle nach.

Neuronale Netze und Statistik weisen große Gemeinsamkeiten auf. Die Statistik beschäftigt sich mit Datenanalyse; in vielen Anwendungen sollen neuronale Netze aus präsentierten Beispielen einen intrinsischen Zusammenhang konstruieren. Wird angenommen, daß die Beispiele zufälligen Einflüssen unterliegen, dann sind neuronale Netze nichts anderes als ein nichtlineares Regressionsmodell in Verkleidung. Kapitel 3 beschreibt, wie diese Sichtweise ausgenutzt werden kann, um die Güte, d. h. die Standardabweichung der Netzausgabe abhängig von der Netzeingabe, des vom neuronalen Netz gefundenen Zusammenhangs mit herkömmlichen Methoden der Statistik zu schätzen. Dies ist von großem Interesse, um das Risiko einer durch die Prognose eines neuronalen Netzes fundierten Entscheidung – etwa bei einer Zeitreihenanalyse – zu beurteilen. Hier ist es besonders wichtig, die Standardabweichung so genau wie möglich angeben zu können. Das ist auch in anderen Bereichen relevant: Beim *aktiven Lernen* werden die Beispiele aus den Bereichen des Eingaberaums ausgewählt, der größtmöglichen weiteren Lernfortschritt verspricht. Ist das Vorgehen dadurch gekennzeichnet, daß Muster ausgesucht werden, bei denen der bisher gelernte Zusammenhang die größte Standardabweichung aufweisen würde, dann ist es natürlich auch wichtig, diese richtig zu schätzen.

Es zeigt sich in einer Reihe von zusammen mit Dr. Arnfried Ossen veröffentlichten Untersuchungen, daß bekannte Methoden der Statistik nicht robuste oder unnötig ungenaue Schätzungen der Standardabweichung geben. Dies kann auf in der Regel durch lokale Optimierungsverfahren verursachte Unzulänglichkeiten neuronalen Lernens zurückgeführt werden. Es gelang uns, mit einer neuen, grundlegenden Idee, solche Unzulänglichkeiten herauszufiltern.

Diese im vierten Kapitel dargelegte Idee ist dann, durch künstlich hinzugefügten Zufall (z. B. für die Wahl der anfänglichen Gewichtungsvektoren des Netzes) die Verteilung der resultierenden Gewichtungsvektoren im Gewichtungsraum zu studieren. Hierbei ist es unerheblich, ob angenommen wurde, daß die Beispiele selbst zufälligen Einflüssen unterliegen oder nicht. Diese Idee ist also auch im allgemeinen Fall, wenn das neuronale Netz kein Regressionsmodell darstellt, gültig. Die Ausführung der Idee

wird zunächst durch offensichtliche Symmetrien behindert: verschiedene Symmetriegruppen wirken auf den Gewichtungsraum und erfordern, daß bestimmte Gewichtungsvektoren miteinander identifiziert werden müssen. Dadurch entsteht in der Regel eine Mannigfaltigkeit, wie in Abb. 2 am Beispiel von durch Gittertranslationen miteinander identifizierten Punkten zu sehen ist.

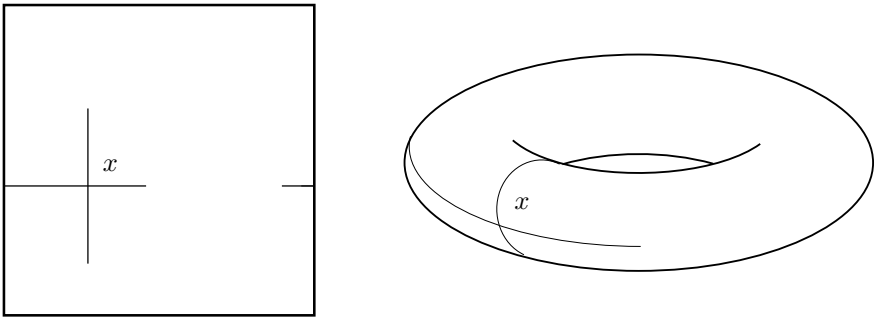


Abb. 2 Von einer Symmetriegruppe erzeugte Mannigfaltigkeit

Nun sind die im Gewichtungsraum wirkenden relevanten Symmetrien keine Translationen, sondern bestimmte Permutationen und Spiegelungen. Ein Algorithmus zur Abbildung von Gewichtungsvektoren in ein nichtredundantes sog. Fundamentalgebiet des Gewichtungsraums wird gegeben. Hier erweist es sich aber als notwendig, die kanonische Metrik auf der so entstandenen Mannigfaltigkeit berechnen zu können. Es stellt sich heraus, daß dies ein in so einer Weise mit dem Problem des Handlungsreisenden verwandtes Problem ist, daß effiziente Approximationen dafür sofort in effiziente Approximationen der Metrik umgesetzt werden können. Dies erlaubt also, Abstände von Gewichtungsvektoren so zu bestimmen, daß sie aussagekräftig für die Ähnlichkeiten der von diesen Gewichtungsvektoren erzeugten Netzfunktionen sind. Herkömmliche Clusteralgorithmen können dann auf der Mannigfaltigkeit Lernergebnisse in Cluster einteilen.

Diese Möglichkeit der Einteilung von Lernergebnissen in Cluster erweist sich als ein Werkzeug mit ungemein vielen Anwendungen: Zum Beispiel können in natürlicher Weise Ensembles von Gewichtungsvektoren visualisiert werden. Im allgemeinen wird das konkrete Netzmodell, welches eine bestimmte Aufgabe lösen soll, dadurch ausgesucht, wie gut das betreffende neuronale Netz es vermag, die vorhandenen Beispiele zu generalisieren. Ein

sehr häufig verwendetes Kriterium für die *Generalisierungsfähigkeit* von neuronalen Netzen ist der sog. Testfehler. Er zeigt in praktischen Anwendungen eine unglaubliche Variabilität, was also eine verläßliche, robuste Modellauswahl deutlich in Frage stellt. Durch Bilden des Testfehlers *pro Cluster* kann eine robuste Beurteilung des Modells stattfinden. Das Bilden einer Pro-Cluster-Statistik ist eine allgemeine Möglichkeit, Statistiken[♮] in bedeutungsvoller Weise robuster zu gestalten: So entstehen schließlich auch jene robusten Schätzungen der Standardabweichung in dem speziellen Fall, daß das neuronale Netz ein Regressionsmodell darstellt.

Neuronale Netze hängen nicht nur mit dem Gebiet der Statistik über die Clusteranalyse als unüberwachtes Lernen, über die Auffassung neuronaler Netze als Regressionsmodell oder über die anfänglichen Verteilungen der Gewichtungsvektoren zusammen, sondern auch über explizite stochastische Netze, wie die Boltzmann-Maschine. Hier gehorchen die Ausgaben der Knoten einer bestimmten, aus dem kanonischen Formalismus der Thermodynamik bekannten Verteilung, der sog. Boltzmann-Gibbs-Verteilung.

Das fünfte Kapitel führt zunächst die bekannte Boltzmann-Maschine aus informationstheoretischer Sicht ein. Will man Boltzmann-Maschinen anwenden, um statistisches Wissen zu speichern oder abzurufen, dann sind beide Prozesse NP-hart, wie weiter im fünften Kapitel argumentiert wird. Die daraus gezogene Lehre ist, daß die Suche nach einer allgemeinen, traktablen Lernregel wenig erfolgversprechend ist.

Die Forschung sollte eher in Richtung spezieller, eingeschränkter Netze gehen. Dies ist Gegenstand des Kapitels 6, des letzten Kapitels. Mittels der aus der statistischen Mechanik bekannten Technik der Dezimierung lassen sich bestimmte Boltzmann-Maschinen leicht behandeln ⟦Saul und Jordan 1994⟧. Diese sind in gewisser Weise eine Verallgemeinerung der aus der Statistik bekannten Hidden-Markow-Modelle, sind daher also relevant ⟦Saul und Jordan 1995, MacKay 1996⟧.

Im sechste Kapitel wird zunächst eine umfassendere Dezimierungsregel vorgestellt; sie erlaubt die effiziente Behandlung einer größeren Klasse von Boltzmann-Maschinen. Es wird des weiteren gezeigt, daß dies schon die allgemeinste Dezimierungsregel ist. Dadurch kann die Menge der *dezimier-*

♮ Eine Statistik ist irgendeine Funktion der zugrundeliegenden Daten.

baren Boltzmann-Maschinen genau charakterisiert werden. Mit Mitteln der Fourier-Stieltjes-Transformation von Wahrscheinlichkeitsmaßen habe ich Lernen und Schließen in dezimierbaren Boltzmann-Maschinen in leicht zu implementierende effiziente Algorithmen übergeführt; die dazu nötigen Techniken der Wahrscheinlichkeitstheorie werden eingeführt. Im wesentlichen ist das Resultat, daß das stochastische und rekurrente Netz einer dezimierbaren Boltzmann-Maschine auf ein deterministisches, nichtrekurrentes Abbildungsnetz zurückgeführt werden kann.

Die entstehenden Algorithmen sind gradientenbasierter Natur, und die von mir im zweiten Kapitel eingeführte stabile Parameteradaption läßt sich zur weiteren Beschleunigung einsetzen. Schließlich kann das Clustern im Gewichtungsraum auch auf Boltzmann-Maschinen angewendet werden, um die Güte einer bestimmten Boltzmann-Maschine, also des Modells, für ein gegebenes Problem zu bewerten. Eine kleine Anwendung demonstriert dies.

1 Backpropagation

In dem Maße, in dem die Komplexität eines Systems ansteigt, nimmt unsere Fähigkeit, präzise und damit signifikante Aussagen über sein Verhalten zu machen, ab. Präzision und Signifikanz schließen sich ab einem gewissen Komplexitätsgrad gegenseitig aus.

— Inkompatibilitätsprinzip (L. A. Zadeh, 1965)

Backpropagation [[z. B. Bryson und Ho 1969, Rumelhart et al. 1986a, le Cun 1989a, McClelland und Rumelhart 1988, Hecht-Nielsen 1989, Müller und Reinhardt 1990, Ritter et al. 1990, Freeman und Skapura 1991, Hertz et al. 1991, Widrow und Lehr 1992, Rojas 1993]] ist eines der bekanntesten Verfahren, um neuronale Netze an ein bestimmtes Eingabe-Ausgabe-Verhalten zu adaptieren.

Zunächst werden in Abschnitt 1.1 allgemeine Abbildungsnetze studiert: Es zeigt sich, daß in natürlicher Weise *zwei* Kettenregeln zum Bilden von Ableitungen gefunden werden, die beide berechtigte, aber verschiedene Anwendungsgebiete haben. Es lohnt sich auch, allgemeinere Netzmodelle zu betrachten: Es sind Shortcuts möglich; Verbindungen können strukturiert weggelassen werden. Weder die Kostenfunktion noch die Aktivierungsfunktionen gehen speziell in die Analyse ein, können also später durch den Anwender nach eigenen Gesichtspunkten frei bestimmt werden. Auch die Wahl, welche Knoten Ausgabeknoten sein sollen, wurde bewußt offen gelassen; dies ist nicht nur nützlich bei der Verallgemeinerung auf rekurrente Netze, sondern auch zum Erzwingen interner Repräsentationen.

Das Theorem von Cybenko (1.2.4) motiviert die Spezialisierung auf neuronale Abbildungsnetze, die aus einfachen, lokalen und parallelen Berechnungselementen bestehen und – so die Aussage – allgemeine Approximatoren sind. Vor diesem Hintergrund wird im darauffolgenden Abschnitt 1.3 der Backpropagation-Algorithmus abgeleitet und dargestellt. Die (nicht

nur affine) Kopplung von Gewichten und der Einsatz günstiger Kostenfunktionen wird in den anschließenden Abschnitten diskutiert.

Als eine Anwendung des möglichst allgemein gehaltenen Rahmens des Backpropagation-Algorithmus werden zwei Lernalgorithmen in rekurrenten Netzen vorgestellt.

1.1 Abbildungsnetze

Ein Netztyp, das sog. Feedforward-Netz, liegt solchen Netzen zugrunde, die einen Eingabevektor eindeutig auf einen Ausgabevektor (im Sinne mathematischer Funktionen) abbilden; im wesentlichen wird das Fehlen von Zyklen verlangt. Dadurch können den einzelnen Knoten des Netzes Abbildungen zugeordnet werden, die ihre Eingabe durch die einlaufenden Kanten bekommen, und deren Ausgabe an alle auslaufenden Kanten kopiert wird. Solche Abbildungsnetze verallgemeinern die Struktur der Hintereinanderausführung vektorwertiger Funktionen; letztere ist durch Abbildungsnetze mit einer Schichtstruktur gegeben.

Feedforward-Netze (1.1.1) Ein *Feedforward-Netz* ist ein endlicher, zyklenfreier gerichteter Graph. Er besteht aus einer endlichen nicht leeren Menge N von *Knoten* und einer Menge $E \subset N \times N$, den *Kanten*, mit der Eigenschaft, daß für alle $x_1, x_2, \ldots x_n \in N$ $(n \geq 2)$ gilt:

$$(x_1, x_1) \notin E$$
$$(x_1, x_2), (x_2, x_3), \ldots, (x_{n-1}, x_n) \in E \Longrightarrow (x_n, x_1) \notin E$$

Ein Spezialfall hiervon ist ein aus disjunkten *Schichten* N_o, ..., N_k, N_{k+1} bestehender Graph mit $N = N_o \cup \ldots \cup N_{k+1}$ und $E = N_o \times N_1 \cup \ldots \cup N_k \times N_{k+1}$. Hier wird die Sprechweise von der *Eingabeschicht* N_o und den k *Binnenschichten* N_1, ..., N_k nützlich werden. □

Beispiel (1.1.2) Das Feedforward-Netz mit $N = \{1, 2, 3, 4, 5, 6\}$ und $E = \{(1, 2), (2, 3), (3, 6), (4, 3), (4, 5), (3, 5)\}$ ist in Abb. 3 gezeichnet. □

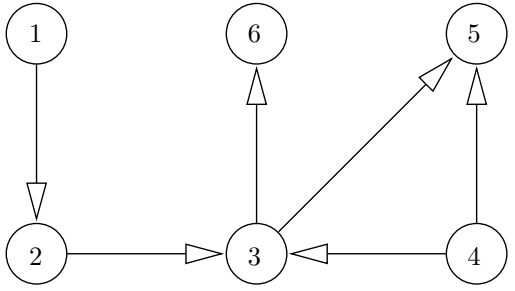


Abb. 3 Beispiel eines Feedforward-Netzes

Topologisches Sortieren (1.1.3) Durch zusätzliche Kanten (a_1, a_n), wann immer es einen gerichteten Weg $\{(a_1, a_2), (a_2, a_3), \ldots (a_{n-1}, a_n)\} \subset E$ gibt, wird aus E eine Halbordnung (die Zyklenfreiheit von (N, E) wird dazu benutzt). Jede Bijektion $x\colon \{1, \ldots, |N|\} \to N$ mit $(x_i, x_j) \in E \Longrightarrow i < j$ heißt *topologische Sortierung* der Knoten *bezüglich* E. Sie ist eine Einbettung der durch E induzierten Halbordnung in eine lineare Ordnung: Wird $x_1, \ldots, x_{|N|}$ von links nach rechts gezeichnet, dann haben alle Kanten von E die gleiche Richtung.

Jede Halbordnung kann topologisch sortiert werden ⟦Szpilrajn 1930⟧.

Die Idee zur topologischen Sortierung eines gerichteten, zyklenfreien, endlichen Graphen (N, E) ist sehr einfach: Gib alle Knoten des Graphen aus, die keinen Vorgänger haben, und lösche die ausgegebenen Knoten aus dem Graphen. Wiederhole diesen Schritt mit dem neuen Graphen bis alle Knoten bearbeitet wurden. □

Beispiel (1.1.4) Für obiges Beispiel (1.1.2) sind $x = (1, 4, 2, 3, 5, 6)$ oder $x = (1, 2, 4, 3, 6, 5)$ topologische Sortierungen bezüglich E. □

Nachbarschaftsbeziehungen in Netzen (1.1.5) Lokal gibt es zu Knoten Nachbarschaftsbeziehungen zu anderen Knoten. Diese sind in der Menge E kodiert. Sei $\Pi_1\colon E \to N$ die Projektion auf die erste und $\Pi_2\colon E \to N$ die Projektion auf die zweite Komponente, z. B. $\Pi_1((a, b)) = a$. Dann ist

$$L_a := \Pi_1(\Pi_2^{-1}(a)),\ a \in N,$$

die Menge der *Vorgängerknoten* von a, d. h. L_a besteht aus den Knoten, von denen aus Verbindungen in Richtung a existieren. Umgekehrt ist

$$R_a := \Pi_2(\Pi_1^{-1}(a))$$

die Menge der *Nachfolgerknoten* eines Knotens. Die sog. *Eingabeschicht*

$$I := N \setminus \Pi_2(E)$$

enthält alle Knoten, die nie als Nachfolgerknoten auftauchen. □

Die Feedforward-Eigenschaft eines Graphen kann dazu benutzt werden, lokale Berechnungen an Knoten durchzuführen und Berechnungsergebnisse entlang der Kanten an Nachfolgerknoten als Eingabe weiterzugeben.

Abbildungsnetze (1.1.6) Ein *Abbildungsnetz* ist ein Feedforward-Netz (N, E) zusammen mit einer Abbildung

$$o_a\colon \begin{cases} \mathbb{R}^{L_a} \to \mathbb{R} & \text{falls } a \in N \setminus I \\ \mathbb{R} \to \mathbb{R} & \text{falls } a \in I \end{cases}$$

pro Knoten a und einer beliebigen, nicht leeren Teilmenge $O \subset N$.

Folgende Schreib- und Redeweisen werden vereinbart. Es heiße

- O die Ausgabeschicht,
- $p \in \mathbb{R}^I$ *Eingabevektor des Netzes* (als freie Variable),
- $h_a \in \mathbb{R}^{L_a}$ *Eingabevektor eines Knotens* $a \in N \setminus I$,
- die Zahl $o_a(h_a)$ bzw. $o_a(p_a)$ auch Aktivation o_a (es muß also dem Kontext entnommen werden, was mit o_a gemeint ist),
- und $y \in \mathbb{R}^O$, $y_a = o_a$ der *Ausgabevektor* des Netzes.

Durch die (induktive) *Vernetzungseigenschaft* für $(a, b) \in E$

$$h_{ba} := (h_b)_a = \begin{cases} o_a(p_a) & \text{falls } a \in I \text{ und} \\ o_a(h_a) & \text{sonst} \end{cases}$$

(kurz „$h_{ab} = o_b$“) wird die sog. *Netzabbildung* induziert:

$$\mathrm{out}\colon \mathbb{R}^I \to \mathbb{R}^O,\ p \mapsto y. \qquad \square$$

Abbildungsnetze kommen also durch nichttriviale Hintereinanderausführung von Abbildungen zustande. Der Eingabevektor h_a eines Knotens a ist für die Abbildung o_a eine freie Variable; jedoch innerhalb des gesamten

Abbildungsnetzes wird h_a durch obige Vernetzungseigenschaft gebunden und hängt von p, dem Eingabevektor des Netzes, ab. Um einen speziellen Wert einer Aktivation o_a oder ihres Eingabevektors h_a im Abbildungsnetz für einem bestimmten Eingabevektor p des Netzes auszudrücken, wird $o_a|_p$ und $h_a|_p$ geschrieben.

Berechnung eines Funktionswertes (1.1.7) Liegt eine topologische Sortierung x der Knoten bezüglich der Kanten vor, so kann der Wert $\mathrm{out}(p)$ der Netzabbildung an der Stelle $p \in \mathbb{R}^I$ berechnet werden, indem mit aufsteigendem i die Werte o_{x_i} bestimmt werden. Dabei wird immer nur auf bereits berechnete Werte zurückgegriffen. $\mathrm{out}(p)$ ordnet dann jedem Knoten von O seine Aktivation zu. □

Bei Abbildungsnetzen gibt es ein einfaches Verfahren, deren *Jacobimatrix* $\mathrm{D\,out} : O \times I \to \mathbb{R}^{\mathbb{R}^I}$

$$\mathrm{D\,out} := (n, a) \mapsto \frac{\partial(\mathrm{out}(p))_n}{\partial p_a}$$

an einer Stelle $p \in \mathbb{R}^I$ zu berechnen. Dazu dient folgendes Lemma:

Lemma: Kettenregel in Abbildungsnetzen (1.1.8) *Gegeben sei ein Abbildungsnetz $(N, E, a \mapsto o_a, O)$, $n \in O$ und ein $p \in \mathbb{R}^I$. Die Abbildungen o_a seien jeweils differenzierbar. Durch die induktive Vorschrift*

$$\delta_i^n := \begin{cases} 1 & \text{für } i = n \\ 0 & \text{falls } R_i = \emptyset \wedge i \neq n \\ \displaystyle\sum_{j \in R_i} \delta_j^n \frac{\partial o_j(h_j)}{\partial h_{ji}} \Big|_{h_j|_p} & \text{sonst} \end{cases}$$

werden jedem Knoten i je $|O|$ Zahlen zugeordnet. Es gilt dann

$$\mathrm{D\,out}\Big|_p = (n, a) \mapsto \delta_a^n \frac{\partial o_a}{\partial p_a}\Big|_p . \qquad \square$$

Beweis (1.1.9) Zunächst kann jedes Abbildungsnetz in Schichten $N_o = I$, N_1, …, N_{k+1} zerlegt werden, so daß für alle $(a, b) \in E$ gilt $a \in N_i$, $b \in N_j$ $\Longrightarrow j > i$. Dann können Identitätsknoten mit dem Ziel eingefügt werden, daß im neuen Netz (N', E') gilt $O \subset N'_{k+1}$ und daß für alle $(a, b) \in E'$ gilt

$a \in N'_i \Longrightarrow b \in N'_{i+1}$. Dadurch verschwinden Shortcuts und die Ausgabeschicht wird in die letzte Schicht transportiert (Abb. 4). Es entsteht ein gleichwertiges Netz, in dem sowohl alle Aktivationen als auch die Werte für δ_i^n für alle $(n,i) \in O \times N$ jeweils mit denen des ursprünglichen Netzes übereinstimmen [Rüger 1996a]. Durch die Zerlegung des Netzes in Schichten wird jedem Knoten a ein eindeutiger *Schichtindex* $s(a)$ zugeordnet, so daß $a \in N_{s(a)}$.

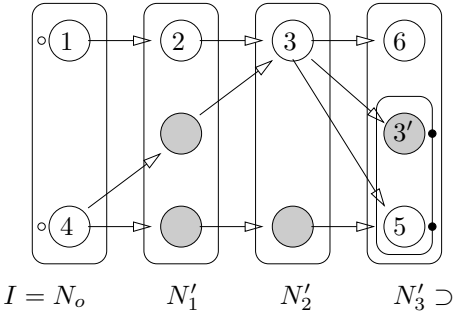


Abb. 4 Schichteinteilung eines Feedforward-Netzes mit Hilfe von Identitätsknoten (grau unterlegt) am Beispiel (1.1.2)

Dann kann o. B. d. A. angenommen werden, daß (N, E, o, O) eine Schichtzerlegung $I = N_o, \ldots, N_{k+1} \supset O$ ohne Shortcuts besitzt. Die Netzfunktion out kann daher als Hintereinanderausführung

$$\begin{aligned} \mathrm{out} : \mathbb{R}^I \xrightarrow{F_o} \mathbb{R}^{N_o} \qquad & \xrightarrow{F_1} \mathbb{R}^{N_1} \qquad \ldots \xrightarrow{F_{k+1}} \mathbb{R}^{N_{k+1}} \\ p \mapsto (a \mapsto f_a(p_a)) \mapsto (F_1 \circ F_o)(p) & \ldots \mapsto (F_{k+1} \circ \dot{\cdots} \circ F_o)(p) \end{aligned}$$

von $k+2$ Funktionen $F_o, \ldots, F_{k+1}$ beschrieben werden. Die Hintereinanderausführung von Abbildungen wird im folgenden als implizites Produkt (unter Weglassen von $\circ$) geschrieben. Für $i < j$ bezeichne $F_i \dot{\cdots} F_j$ die *Identität* $\mathrm{id} : x \mapsto x$ auf $\mathbb{R}^{N_j}$.

Sei n nun beliebig, aber fest. Wäre folgende Behauptung über δ_i^n

$$\delta_i^n = \mathrm{D}\, F_{s(n)} \dot{\cdots} F_{s(i)+1} \Big|_{F_{s(i)} \dot{\cdots} F_o(p)} (n,i) \qquad (*)$$

für alle $i \in N$ bewiesen, so folgt daraus unmittelbar (nach einmaliger Anwendung der Kettenregel) die Behauptung des Lemmas.

Nun ist $(*)$ sicher richtig für alle i mit $s(i) \geq s(n)$. D wirkt auf die Identität, und deren Jacobimatrix ist die Einheitsmatrix; dies stimmt genau mit der Definition der Basisfälle von δ_i^n überein.

Sei $(*)$ nun schon für alle $i \in N_l$ mit $0 < l \leq s(n)$ gezeigt; betrachte dann für $i \in N_{l-1}$

$$\begin{aligned}
&\mathrm{D}F_{s(n)} \cdot\!\cdot\!\cdot F_l \Big|_{F_{l-1} \cdot\!\cdot\!\cdot F_o(p)} (n,i) \\
&= \left(\mathrm{D}\, F_{s(n)} \cdot\!\cdot\!\cdot F_{l+1} \Big|_{F_l \cdot\!\cdot\!\cdot F_o(p)} \cdot \mathrm{D}\, F_l \Big|_{F_{l-1} \cdot\!\cdot\!\cdot F_o(p)} \right) (n,i) \\
&= \sum_{j \in N_l} \left(\mathrm{D}\, F_{s(n)} \cdot\!\cdot\!\cdot F_{l+1} \Big|_{F_l \cdot\!\cdot\!\cdot F_o(p)} (n,j) \cdot \mathrm{D}\, F_l \Big|_{F_{l-1} \cdot\!\cdot\!\cdot F_o(p)} (j,i) \right) \\
&= \sum_{j \in N_l} \delta_j^n \, \mathrm{D}\, F_l \Big|_{F_{l-1} \cdot\!\cdot\!\cdot F_o(p)} (j,i) \\
&= \sum_{j \in R_i} \delta_j^n \frac{\partial o_j(h_j)}{\partial h_{ji}} \Big|_{h_j|_p} \\
&= \delta_i^n .
\end{aligned}$$

Der erste Schritt ist die Anwendung der Kettenregel, der zweite eine Matrixmultiplikation, der dritte verwendet die Induktionsannahme $(*)$ für $j \in N_l$ und schließlich wird die Struktur von F_l ausgenutzt. Damit aber ist genau der induktive Teil der Definition von δ_i^n erreicht. ∎

Beispiel (1.1.10) Die Kettenregel für Abbildungsnetze führt rekursiv die Deltas auf andere zurück. Stellt man sich δ_i^n als Matrixelemente einer $|O| \times |N|$-Matrix vor, dann werden Deltas einer Zeile auf Deltas nur dieser Zeile zurückgeführt.

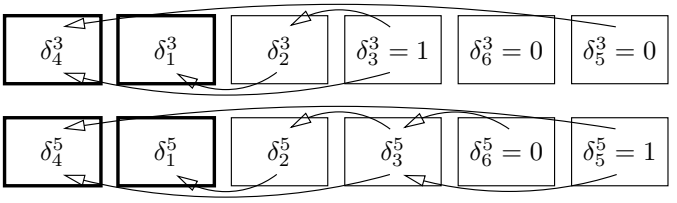


Abb. 5 Beispiel für die rekursive Abhängigkeit der Deltas

Abb. 5 veranschaulicht die Art der Rekursion für das Feedforward-Netz aus Abb. 3 mit $O = \{3, 5\}$: die Pfeile, welche die Abhängigkeiten kennzeichnen, sind zu Kanten assoziiert. Sind nun innerhalb des Zeilenvektors δ^n seine Indizes topologisch sortiert, dann zeigen alle Pfeile (und damit die Induktionsrichtung) von rechts nach links. Und links stehen auch die für die Jacobimatrix benutzten Deltas (dick eingerahmt). □

Alternative Kettenregel in Abbildungsnetzen (1.1.11) *Es gibt andere Möglichkeiten, die Jacobimatrix einer Netzabbildung zu berechnen. Ordne z. B. jedem Knoten $n \in N$ für jeden Eingabeknoten $i \in I$ folgende induktiv definierte Zahl zu:*

$$\gamma_i^n := \begin{cases} \frac{\partial o_i}{\partial p_i}\Big|_p & \text{für } n = i \\ 0 & \text{falls } n \in I \setminus \{i\} \\ \sum_{m \in L_n} \gamma_i^m \frac{\partial o_n(h_n)}{\partial h_{nm}}\Big|_{h_n|_p} & \text{sonst} \end{cases}$$

Es gilt dann $D\,\mathrm{out}\,|_p(n, i) = \gamma_i^n$ für alle $(n, i) \in O \times I$.

Der Rechenaufwand eines hieraus abgeleiteten Algorithmus ist nicht notwendigerweise gleich zu dem des Lemmas (1.1.8). □

Beweis (1.1.12) Der erste Teil des Beweises ist völlig analog zum Beweis (1.1.9) vom obigen Lemma, nur wird beim Anwenden der Kettenregel der linke statt des rechten Faktors abgespalten und dadurch eine umgekehrte Induktionsrichtung benutzt.

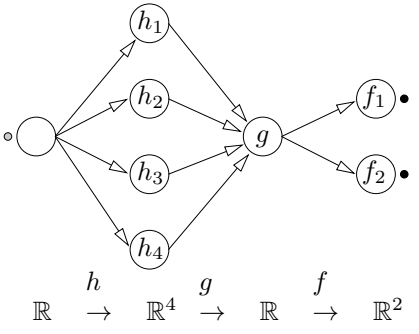


Abb. 6 Abbildungsnetz für eine Funktion $f \circ g \circ h$

Zum Rechenaufwand: Sei $f\colon \mathbb{R} \to \mathbb{R}^2$, $g\colon \mathbb{R}^4 \to \mathbb{R}$ und $h\colon \mathbb{R} \to \mathbb{R}^4$; dann kann die Hintereinanderausführung $f \circ g \circ h$ in natürlicher Weise als Abbildungsnetz geschrieben werden (siehe Abb. 6). Beide Kettenregeln für Abbildungsnetze implementieren in diesem Spezialfall das Matrixprodukt $Df \cdot Dg \cdot Dh$ der regulären Kettenregel, nur die Reihenfolge der Matrixmultiplikationen unterscheidet sich. Nun sind Matrixmultiplikationen zwar assoziativ, aber die Anzahl der elementaren Rechenoperationen kann unterschiedlich sein (siehe Abb. 7). ∎

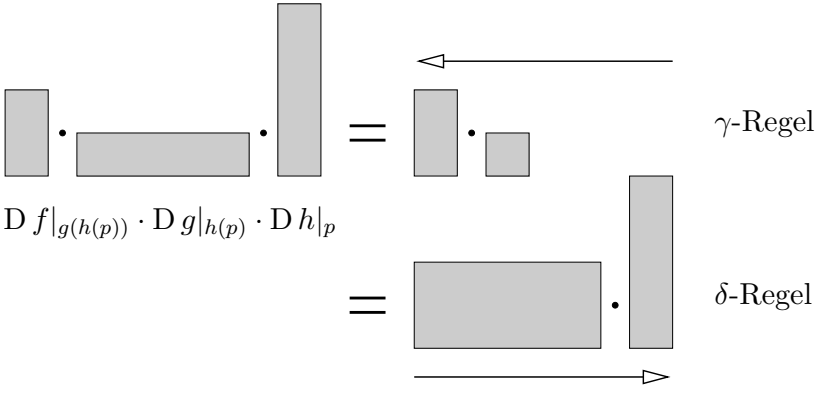


Abb. 7 Induktionsrichtung der Kettenregel für Abbildungsnetze

Es wird sich in (1.3.5) zeigen, daß die δ-Kettenregel bei Feedforward-Netzen von Vorteil ist, jedoch die γ-Kettenregel wegen ihrer Induktionsrichtung bei rekurrenten Backpropagation-Netzen (Kapitel 1.6) sehr gut zum Einsatz kommen kann.

1.2 Neuronale Abbildungsnetze

Abbildungsnetze sind sehr allgemein; jede Abbildung $g\colon \mathbb{R}^n \to \mathbb{R}^m$ kann in kanonischer Weise als zweischichtiges Abbildungsnetz mit n Identitätsknoten in der Eingabeschicht und m Ausgabeknoten $1, \ldots, m$ aufgefaßt werden. Die Aktivation o_i für den Knoten $1 \leq i \leq m$ ist gerade die i-te Komponente von g. Im Sinne einfacher Berechnungselemente ist es jedoch nicht praktikabel, beliebige Abbildungen von $\mathbb{R}^n \to \mathbb{R}$ als Aktivationen zuzulassen.

Es ist viel einfacher, am Knoten a eine eindimensionale Abbildung f_a nach einer linearen Abbildung $\mathbb{R}^{L_a} \to \mathbb{R}$ anzuwenden. Lineare Abbildungen nach $\mathbb{R}$ sind aber gerade die Skalarmultiplikationen mit Vektoren aus $\mathbb{R}^{L_a}$. Dadurch ist die Aktivation o_a mit $|L_a|$ Zahlen parametrisierbar. Abbildungsnetze, deren Aktivationen sich auf diese einfache Struktur beschränken, können immer noch eine große Klasse von Funktionen darstellen, wie sich im folgenden zeigen wird.

Neuronale Abbildungsnetze (1.2.1) Ein *neuronales Abbildungsnetz* ist ein Feedforward-Netz (N, E) mit $E \neq \emptyset$ zusammen mit

- einer Abbildung $w\colon E \to \mathbb{R}$, dem sog. *Gewichtungssatz* (statt $w_{(i,j)}$ wird auch w_{ij} geschrieben; diese Zahlen heißen Gewichte),
- *Aktivierungsfunktionen* $f\colon N \to \mathbb{R}^{\mathbb{R}}$, $a \mapsto f_a\colon \mathbb{R} \to \mathbb{R}$ und
- einer beliebigen nicht leeren Menge $O \subset N$, der Ausgabeschicht.

Mit der Definition von

$$a \mapsto o_a = \begin{cases} h_a \mapsto f_a\left(\sum_{n \in L_a} h_{an} w_{na}\right) & \text{falls } a \in N \setminus I \\ p_a \mapsto f_a\left(p_a\right) & \text{sonst} \end{cases}$$

pro Knoten a erhält damit (N, E, f, w, O) die Struktur einer Schar von speziellen Abbildungsnetzen. Demzufolge trägt die Netzfunktion $\mathrm{out}_w\colon \mathbb{R}^I \to \mathbb{R}^O$ den Index w. Abb. 8 veranschaulicht obige Begriffe. Das Argument einer Aktivierungsfunktion am Knoten $a \in N \setminus I$ heißt auch *Nettoeingabe* des Knotens a: die Nettoeingabe ist die gewichtete Summe der Aktivationen von Vorgängerknoten. □

Bias (1.2.2) Bisher wurden noch keine Annahmen über die Art der Aktivierungsfunktion eines Knotens, z. B. n, gemacht. Häufig wird jedoch eine Abbildung verwendet, die i. w. den negativen Zahlen Null (oder kleine positive Zahlen) und den positiven Zahlen Eins (oder etwas weniger) zuordnet. So eine Aktivierungsfunktion diskriminiert i. w. die beiden Definitionsbereiche $(-\infty, b_n]$ und $[b_n, \infty)$ mit $b_n = 0$. Um einen anderen Arbeitspunkt $b_n \neq 0$ dieser Aktivierungsfunktion zu erreichen, wird oft b_n als zusätzlicher Parameter, der sog. *Schwellenwert* (auch minus *Bias*), des Knotens eingeführt und $x \mapsto f_n(x - b_n)$ als Aktivierungsfunktion verwendet.

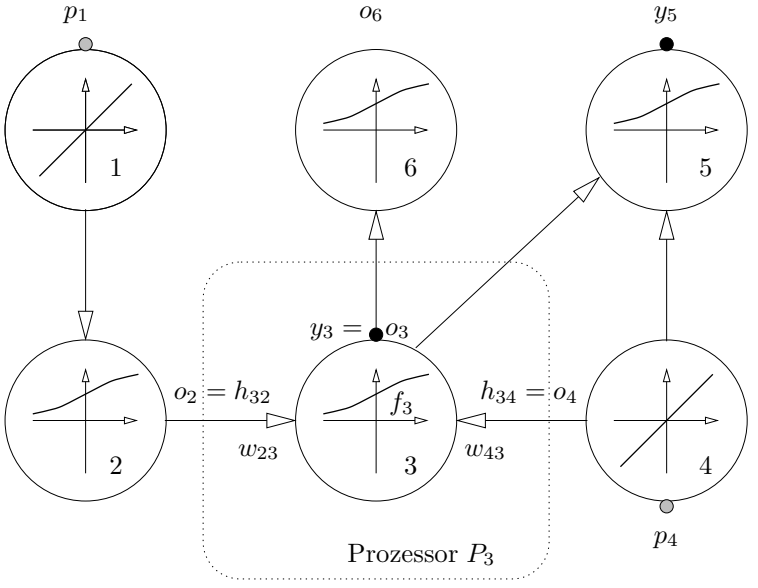


Abb. 8 Ein neuronales Abbildungsnetz $(p_1, p_4) \overset{\text{out}_w}{\longmapsto} (y_3, y_5)$ mit $O = \{3, 5\}$

Solche Schwellenwerte sind in neuronalen Abbildungsnetzen aber bereits als reguläre Gewichte modellierbar: Mit einem weiteren Knoten $0 \in N$ mit Verbindungen zu allen bisherigen Nichteingabe-Knoten, $(0, a) \in E$ falls $a \notin I$, einer konstanten Aktivierungsfunktion $f_o\colon \mathbb{R} \to \mathbb{R}$, $x \mapsto 1$, fungiert das Gewicht $-w_{oa}$ als Schwellenwert des Knotens a. Der neue Knoten 0 gehört damit formal zur Eingabeschicht; seine Eingabe p_o hat jedoch keinen Einfluß auf die Netzausgabe y und wird daher nicht wirklich als Eingabevariable gezählt. Der *Biasknoten* 0 wird als $\dashv\!\Vdash$ gezeichnet. □

Netze als lokale Prozessoren (1.2.3) Ein neuronales Abbildungsnetz (N, E, w, f, O) kann auch aufgefaßt werden als lokale Verschaltung von *Prozessoren* P_a, $a \in N$, zu denen je eine Menge L_a von Vorgängerknoten, eine Menge R_a von Nachfolgerknoten, ein Eingabevektor $h_a \in \mathbb{R}^{L_a}$, ein Gewichtungsvektor $w_{\bullet a} \in \mathbb{R}^{L_a}$ und eine Aktivierungsfunktion f_a gehört. Ist L_a leer, so gehört der Prozessor zur Eingabeschicht und benötigt keinen Gewichtungsvektor; anstelle des zugehörigen Eingabevektors tritt dann eine unabhängige Eingabevariable p_a. Wie in (1.2.1) kann jedem Prozessor eine Aktivation zugewiesen werden; die Verschaltung der Prozessoren erfolgt dann mit Hilfe der Vernetzungseigenschaft. Prozessoren einer Schicht

können dann ihre Ergebnisse parallel berechnen, indem sie auf Berechnungsergebnisse der Prozessoren früherer Schichten zurückgreifen. □

$w \mapsto \text{out}_w$ ist eine ganze Schar von Funktionen; je nach Gewichtungssatz wird eine andere Funktion $\mathbb{R}^I \to \mathbb{R}^O$ realisiert. Ein bedeutendes Resultat zur Approximation stetiger Funktionen auf *kompakten* (in $\mathbb{R}^n$: abgeschlossenen und beschränkten) Gebieten ist im folgenden Theorem gegeben.

Theorem von Cybenko (1.2.4) *Sei $\sigma\colon \mathbb{R} \to \mathbb{R}$ eine stetige und sigmoide (d. h. $\lim_{x\to\infty} \sigma(x) = 1$ und $\lim_{x\to-\infty} \sigma(x) = 0$) Funktion. Dann sind endliche Summen der Form*

$$G(x) = \sum_{j=1}^{K} \alpha_j \sigma\left(y_j x + \theta_j\right), x \in \mathbb{I}^n,$$

mit den Parametern $\alpha_j, \theta_j \in \mathbb{R}$ und $y_j \in \mathbb{R}^n$ dicht im Raum $C(\mathbb{I}^n)$ der stetigen reellwertigen Funktionen auf dem abgeschlossenen Einheitswürfel $\mathbb{I}^n := [0,1]^n$ bezüglich der Supremumsnorm.

Mit anderen Worten: sei eine Funktion $f \in C(\mathbb{I}^n)$ gegeben und ein $\varepsilon > 0$, dann gibt es ein K und Parameter $\alpha_j, \theta_j \in \mathbb{R}$ und $y_j \in \mathbb{R}^n$ $(1 \le j \le K)$, so daß für alle $x \in \mathbb{I}^n$ gilt

$$|G(x) - f(x)| < \varepsilon.$$ □

Beweis (1.2.5) Ein Beweis obigen Theorems wurde in ⟦Cybenko 1989⟧ gegeben. Er benutzt Methoden der Funktionalanalysis und der Maßtheorie (u. a. Hahn-Banach-Theorem und Rieszsches Darstellungstheorem). ■

Relevanz für neuronale Abbildungsnetze (1.2.6) Dieses Theorem kann sofort auf neuronale Abbildungsnetze angewendet werden: Betrachte ein Netz mit n Eingabeknoten, einer Binnenschicht mit einer endlichen (aber unbestimmten) Zahl K von Knoten und einem Ausgabeknoten. Die Aktivierungsfunktionen von Knoten der Binnenschicht seien alle durch eine sigmoide Funktion σ gegeben, die von Knoten der Eingabeschicht sowie der Ausgabeschicht durch die Identität. Die Gewichte zu einlaufenden Kanten des j-ten Binnenknoten seien durch den Vektor y_j gegeben, sein Bias durch die Zahl θ_j und das Gewicht zum Ausgabeknoten durch α_j.

Die Netzfunktion eines solchen Netzes kann dann mit der Funktion G des Theorems von Cybenko identifiziert werden. Gegeben eine beliebige Funktion f von $\mathbb{I}^n \to \mathbb{R}$ und eine beliebige Fehlerschranke ε; dann gibt es also ein K und ein so dimensioniertes neuronales Abbildungsnetz mit einer bestimmten Gewichtungsbelegung, so daß dessen Netzfunktion im gesamten Definitionsbereich innerhalb eines ε-Schlauchs um f liegt.

Durch Parallelverschaltung (siehe Abb. 9) von m Netzen mit durchaus verschieden dimensionierten Binnenschichten kann demnach jede stetige Funktion von $\mathbb{I}^n$ nach $\mathbb{R}^m$ approximiert werden.

Darin liegt die enorme Bedeutung des Theorems von Cybenko. □

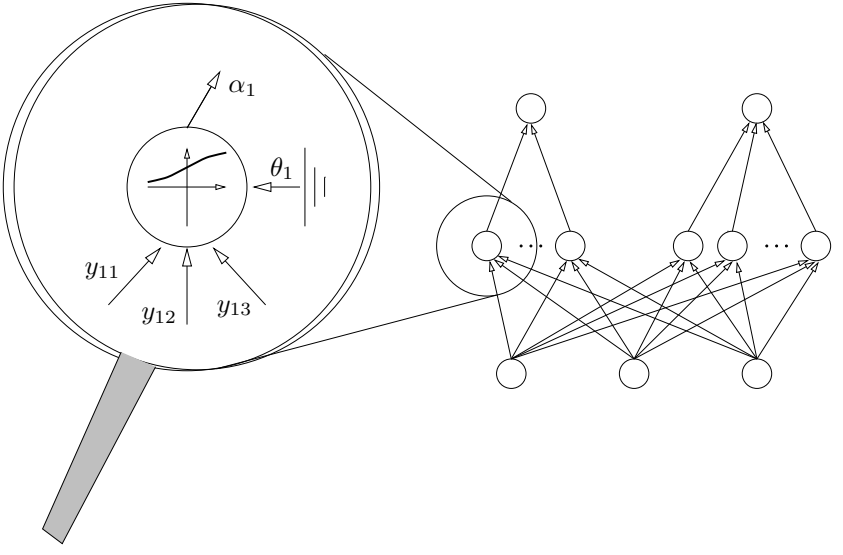


Abb. 9 Beispielnetz zur Approximation einer Funktion $f\colon \mathbb{I}^3 \to \mathbb{R}^2$

Bemerkungen (1.2.7) (i) Manchmal wird unter dem Begriff sigmoid auch Monotonie subsummiert. Das ist hier explizit nicht der Fall.

(ii) Ein ähnliches Approximationstheorem wurde unabhängig auch von Hornik, Stinchcombe und White [[Hornik et al. 1989]] gefunden. Verschiedenste Variationen in der Voraussetzung obigen Theorems wurden durchgespielt: es wurde u. a. gezeigt, daß statt stetigen und sigmoiden Aktivierungsfunktionen auch monotone und sigmoide zum selben Approximations-

ergebnis führen. Dies gibt dem Anwender neuronaler Netze große Freiheit in der Wahl der Aktivierungsfunktionen.

(iii) Die Kompaktheit des Definitionsbereichs der zu approximierenden Funktionen geht jedoch wesentlich in alle Beweise ein. □

Die Definition der neuronalen Abbildungsnetze ist mächtig genug, um mit solchen beliebige stetige Funktion $\mathbb{I}^n \to \mathbb{R}^m$ bezüglich der Supremumsnorm zu approximieren. Bisher konnte allerdings nur die prinzipielle Lösbarkeit der Aufgabe zugesichert werden.

Die Suche nach einem Verfahren zur Lösung des Approximationsproblems ist eine breite Spielwiese mit vielen offenen Fragen. Beispielsweise ist die Wahl der Aktivierungsfunktionen in einem weiten Spektrum beliebig; abhängig von der gewählten Aktivierungsfunktion und dem gestellten Problem ist die Frage nach der geometrischen Struktur (N, E) i. allg. nicht einfach zu beantworten. Selbst wenn im Einzelfall diese geklärt ist, besteht in der Wahl des Gewichtungssatzes w noch ein nichttriviales Teilproblem.

Die Methode des Gradientenabstiegs (1.2.9) ist ein Ansatz von vielen, dieses Teilproblem bei einem ansonsten festen Netz zur Approximation einer vorgegebenen Abbildung g zu lösen.

Backpropagation-Netze (1.2.8) Ein *Backpropagation-Netz* ist ein neuronales Abbildungsnetz mit differenzierbaren Aktivierungsfunktionen zusammen mit einer differenzierbaren Abbildung $d\colon \mathbb{R}^O \times \mathbb{R}^O \to \mathbb{R}$, die eine Abweichung zweier Elemente von $\mathbb{R}^O$ angibt, z. B. $d(x, y) := (x-y)^2/2$ (im Sinne eines Skalarprodukts). d heißt auch *Kostenfunktion*; sie soll nach unten beschränkt sein und für jedes x soll die Abbildung $y \mapsto d(x, y)$ bei $y = x$ ihr Minimum annehmen.

Sei $S \subset \mathbb{R}^I \times \mathbb{R}^O$ eine *Pattern-Target-Relation*, bestehe also aus Beispielen von *Eingabe-Ausgabe-Mustern* $(p, t) \in \mathbb{R}^I \times \mathbb{R}^O$ (ein typisches S wäre der Graph einer zu approximierenden Funktion $g\colon \mathbb{R}^I \to \mathbb{R}^O$). d induziert dann auf S einen *Einzelfehler* $\mathrm{err}_w : S \to \mathbb{R}$, $(p, t) \mapsto d\,(t, \mathrm{out}_w(p))$. Der Einzelfehler gibt für ein Eingabe-Ausgabe-Muster an, wie gut (oder schlecht) das Backpropagation-Netz dieses wiedergeben kann.

Der Gesamtfehler der Approximation von S wird als Summe der Einzelfehler der verwendeten Trainingsmuster aus einer endlichen Teilmenge $T \subset S$

angesehen. Für ein konkretes neuronales Abbildungsnetz hängt bei festem N, E und f der Gesamtfehler nur noch von der Wahl der Gewichte ab. Dieser Sachverhalt motiviert folgende Definition des *Gesamtfehlers*:

$$\begin{aligned} \mathrm{Err}_T : \mathbb{R}^E &\to \mathbb{R}_o^+ \\ w &\mapsto \sum_{(p,t)\in T} \mathrm{err}_w(p,t) \end{aligned}$$ □

Methode des Gradientenabstiegs (1.2.9) Es liegt nahe, zur Approximation mit einem Backpropagation-Netz ein solches $w^* \in E$ zu suchen, bei dem $w \mapsto \mathrm{Err}_T(w)$ ein (absolutes) Minimum annimmt. Eine Möglichkeit der Suche ist dann der sog. *Gradientenabstieg*: Wähle $w^o \in \mathbb{R}^E$ zufällig oder beliebig. Setze für $i > 0$

$$w^i = w^{i-1} - \eta \nabla \mathrm{Err}_T \Big|_{w^{i-1}} \quad \text{mit } \eta \in \mathbb{R}^+ \text{ klein.}$$

Die Hoffnung ist, daß die so entstehende Folge bei einem lokalen Minimum zur Stagnation kommt. Err_T ist nach unten beschränkt, und – wenn die *Lernrate* η nur klein genug ist, der Betrag des Gradienten von Err_T beschränkt ist und w^o nahe genug an einem Minimum ist – die Folge $i \mapsto \mathrm{Err}_T(w^i)$ monoton fallend, also konvergent. $i \mapsto w^i$ kommt an einem lokalen Minimum, einem Sattelpunkt oder gar einem lokalen Maximum (wenn w^o schon ein lokales Maximum war) zum Stillstand. □

1.3 Der Backpropagation-Algorithmus

Backpropagation ist ein Algorithmus zur Implementierung der Methode des Gradientenabstiegs. Seine Stärke liegt in der effizienten Benutzung der lokalen Prozessoren zur Berechnung von $\nabla(w \mapsto \mathrm{err}_w(p,t))$. Die Summation über $(p,t) \in T$ ergibt dann den erwünschten Gradienten $\nabla \mathrm{Err}_T$.

Backpropagation-Theorem (1.3.1) *Sei ein Backpropagation-Netz* (N, E, f, w, O, d) *gegeben. Bilde für* $a \in N \setminus I$ *und* $(p,t) \in \mathbb{R}^I \times \mathbb{R}^O$

$$\delta_a := d_a \cdot \left(\sum_{b \in R_a} \delta_b w_{ab} + \frac{\partial d(t,y)}{\partial y_a} \Big|_{y=\mathrm{out}_w(p)} \right),$$

wobei die Summation über $b \in R_a$ als 0 erklärt ist, wenn R_a leer ist, und die partielle Ableitung als 0 erklärt ist, wenn $a \notin O$; d_a sei als Abkürzung

$$d_a := f_a' \Big|_{\sum_{j \in L_a} h_{aj} w_{ja}} \quad \text{für } a \in N \setminus I,$$

d. h. als Ableitung f_a' der Aktivierungsfunktion an der Stelle ihrer Nettoeingabe erklärt. Dann ist δ_a für alle $a \in N \setminus I$ wohldefiniert und es gilt

$$\nabla(w \mapsto \mathrm{err}_w(p,t)) \Big|_w = (a,b) \mapsto o_a|_p \, \delta_b. \qquad \square$$

Beweis (1.3.2) Die δ_a sind induktiv entgegen der topologischen Reihenfolge ihrer Knoten definiert, denn es wird nur auf δ_b mit $b \in \mathbb{R}_a$ zurückgegriffen. Die Wohldefiniertheit der Definition ergibt sich daraus, daß die Induktionsverankerung bei Knoten startet, die eine leere Nachfolgermenge besitzen. In einem Feedforward-Netz gibt es wegen der Zyklenfreiheit aber immer Knoten a mit $R_a = \emptyset$.

Bilde nun aus dem Backpropagation-Netz unter Hinzufügen eines Eingabeknotens pro Gewicht und eines Knotens e für den Fehler (siehe Abb. 10 für das Netz aus Abb. 8) ein Abbildungsnetz mit der Netzabbildung

$$\begin{aligned} \mathbb{R}^E \times \mathbb{R}^I &\to \mathbb{R} \\ (w,p) &\mapsto \mathrm{err}_w(p,t). \end{aligned}$$

Dieses Netz hat genau einen Ausgabeknoten e. Durch Anwendung von Lemma (1.1.8) auf dieses Abbildungsnetz und Identifikation von $\delta_a = d_a \cdot \delta_a^{\mathrm{e}}$ ergibt sich sofort die Aussage des Theorems. ∎

Backpropagation-Algorithmus (1.3.3) Sei x eine topologische Sortierung eines Backpropagation-Netzes (N, E, f, w, O, d). Sei eine endliche Menge $T \subset S$ von Eingabe-Ausgabe-Mustern einer zu approximierenden Relation $S \subset \mathbb{R}^I \times \mathbb{R}^O$ gegeben. Folgender Algorithmus implementiert die Methode des Gradientenabstiegs mit Lernrate $\eta \in \mathbb{R}^+$.

Wesentliche Variablen: $w, v \in \mathbb{R}^E$, $o \in \mathbb{R}^N$, $d, \delta \in \mathbb{R}^{N \setminus I}$ und $e, r, n \in \mathbb{R}$.

(i) Initialisierung: Bilde einen anfänglichen Gewichtungssatz w mit betragsmäßig kleinen (z. B. normalverteilten) Werten. Lösche n.

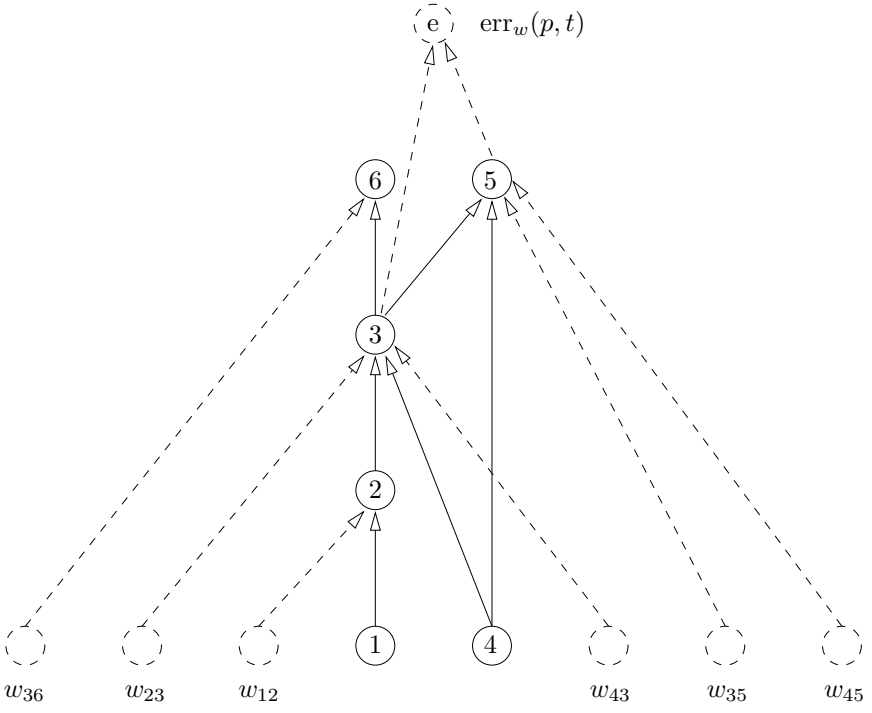


Abb. 10 Beispiel eines Abbildungsnetzes für den Einzelfehler

(ii) Lösche v und e; inkrementiere n. Für jedes $(p,t) \in T$ führe aus:

- Berechne in topologischer Reihenfolge, d. h. mit aufsteigendem $i \leftarrow 1,\ldots,|N|$, die Werte o_{x_i} und d_{x_i} (Propagation).
  - Falls $x_i \in I$ setze $o_{x_i} \leftarrow f_{x_i}(p_{x_i})$, ansonsten
  - berechne $r \leftarrow \sum_{a \in L_{x_i}} o_a w_{ax_i}$, $o_{x_i} \leftarrow f_{x_i}(r)$ und $d_{x_i} \leftarrow f'_{x_i}(r)$.
- Optional: Setze $e \leftarrow e + d(t, \mathrm{out}_w(p))$ (Gesamtfehler).
- Berechne in entgegengesetzter topologischer Reihenfolge, d. h. mit fallendem $i \leftarrow |N|,\ddots,1$, die Werte δ_{x_i} (Backpropagation). Es ist nichts zu tun falls $x_i \in I$, ansonsten lösche r,
  - setze $r \leftarrow r + (\partial d(t,y)/\partial y_{x_i})\,|_{\mathrm{out}_w(p)}$ falls $x_i \in O$
    (oft ist $d(t,y) = (t-y)^2/2$ und dann gilt $r \leftarrow r + o_{x_i} - t_{x_i}$),
  - setze $r \leftarrow r + \sum_{b \in R_{x_i}} \delta_b w_{x_i b}$ falls $R_{x_i} \neq \emptyset$,
  - und setze $\delta_{x_i} \leftarrow d_{x_i} r$.

- Setze $v_{ab} \leftarrow v_{ab} + o_a \delta_b$ für jedes $(a,b) \in E$ (Änderung von v).

(iii) Setze $w \leftarrow w - \eta v$. Ist nun der Betrag des berechneten Gradienten v (oder optional der Gesamtfehler e) klein genug oder die verstrichene Zeit bzw. die Zahl n der Iterationen groß genug, so terminiere. Ansonsten wiederhole obige Schritte ab (ii). □

Backpropagation-Alternative (1.3.4) Der Name Backpropagation kommt von der Induktionsrichtung beim Bestimmen des Gradienten mit Hilfe des Backpropagation-Theorems (1.3.1). Die gängige Redeweise ist die, daß die Aktivationen im Netz *propagiert*, die Fehler (oder auch die Deltas) *zurückpropagiert* werden.

Wird jedoch im Beweis (1.3.2) des Backpropagation-Theorems statt der δ-Kettenregel die γ-Kettenregel (1.1.11) angewendet, so dreht sich die Induktionsrichtung um, und wir erhalten folgende Aussage:

Sei (N, E, f, w, O, d) ein Backpropagation-Netz. Bilde für $n \in N$, $(a,b) \in E$ und $(p,t) \in \mathbb{R}^I \times \mathbb{R}^O$

$$\gamma_{ab}^n := \begin{cases} 0 & \text{falls } L_n = \emptyset, \\ d_n o_a & \text{falls } b = n, \\ d_n \sum\limits_{j \in L_n} \gamma_{ab}^j w_{jn} & \text{sonst.} \end{cases}$$

Dann ist der Gradienten des Einzelfehlers gegeben durch

$$\nabla(w \mapsto \mathrm{err}_w(p,t)) \Big|_w (a,b) = \sum_{n \in O} \gamma_{ab}^n \frac{\partial d(t,y)}{\partial y_n} \Big|_{\mathrm{out}_w(p)}.$$

Dieses Ergebnis kann beinahe wortgetreu in einen zu (1.3.3) bezüglich des Rechenergebnisses äquivalenten Algorithmus umgesetzt werden:

Wesentliche Variablen: $w, v \in \mathbb{R}^E$, $o \in \mathbb{R}^N$, $\gamma \in \mathbb{R}^{E \times N}$ und $e, r, n \in \mathbb{R}$.

(i) Initialisiere w; lösche n.

(ii) Lösche v und e; inkrementiere n. Für jedes $(p,t) \in T$ führe aus:

- Berechne in topologischer Reihenfolge die Werte o_{x_i} und $\gamma_\bullet^{x_i}$:
  – Falls $x_i \in I$ setze $o_{x_i} \leftarrow f_{x_i}(p_{x_i})$,

– ansonsten berechne $r \leftarrow \sum_{a \in L_{x_i}} o_a w_{ax_i}$ und $o_{x_i} \leftarrow f_{x_i}(r)$;

– Setze für alle $(a,b) \in E$

$$\gamma_{ab}^{x_i} \leftarrow \begin{cases} 0 & \text{falls } x_i \in I \\ f'_{x_i}(r) o_a & \text{falls } x_i = b \\ f'_{x_i}(r) \sum\limits_{j \in L_{x_i}} \gamma_{ab}^{j} w_{jx_i} & \text{sonst.} \end{cases}$$

- Optional: Setze $e \leftarrow e + d(t, \mathrm{out}_w(p))$ (Gesamtfehler).
- Setze $v_{ab} \leftarrow v_{ab} + \sum_{n \in O} \gamma_{ab}^{n} \left(\partial d(t,y)/\partial y_n\right)|_{\mathrm{out}_w(p)}$ für jedes $(a,b) \in E$ (Änderung von v).

(iii) Setze $w \leftarrow w - \eta v$. Prüfe eine geeignete Abbruchbedingung und wiederhole ggf. obige Schritte ab (ii). □

Rechenaufwand (1.3.5) Die beiden Algorithmen (1.3.3) und (1.3.4) unterscheiden sich nicht in der äußeren Schleife über die Muster. Deswegen werden die folgenden Zeitbetrachtungen pro Muster durchgeführt. Die Anzahl der Funktionsaufrufe von f_b (das sind $|N|$) bzw. f'_b ($|N \setminus I|$) ist darüberhinaus bei beiden Varianten gleich. Sei m die Zeit für eine Multiplikation und a die Zeit für eine Addition. Die Zeit T, die obiger Algorithmus (1.3.4) pro Muster für Multiplikation und Addition benötigt, kann aufgeteilt werden in Zeit a) zur Berechnung der Aktivationen o_{x_i} , b) zur Berechnung der Gammas $\gamma_{\bullet}^{x_i}$ und c) zur Änderung von v:

$$T = \sum_{x_i \in N \setminus I} \Big(|L_{x_i}|(m+a) \qquad (a)$$

$$+ \big(m + |L_{x_i}|(m+a)\big)(|E| - |L_{x_i}|) + |L_{x_i}|m \Big) \qquad (b)$$

$$+ |E|\,|O|(m+2a) \qquad (c)$$

Dabei wird von der gängigen Programmierpraxis ausgegangen, bei der Berechnung von $\sum_{i \in N} r_i s_i$ eine Variable zu löschen und in einer Schleife sowohl $|N|$ Multiplikationen als auch $|N|$ Additionen auszuführen. Unter

Benutzung von $|E| = \sum_{b \in L_b} |L_b|$ kann T vereinfacht werden zu

$$|E|\Big(\big((|N \setminus I| + |O| + 1)m + (2|O| + 1)a + |E|(m + a)\Big(1 - \sum_b \frac{|L_b|^2}{|E|^2}\Big)\Big).$$

In großen heterogenen Netzen kann die Summe über $(|L_b|/|E|)^2$ vernachlässigt werden. Der generelle Rechenaufwand liegt dann in der Größenordnung von $|E|(|E| + |N|)$.

Hingegen gilt für den Backpropagation-Algorithmus (1.3.3), daß

$$\begin{aligned} T = \sum_{x_i \in N \setminus I} \Big(& |L_{x_i}|(m + a) && (a) \\ & + |R_{x_i}|(m + a) + m \Big) + 2|O|a && (b) \\ & + |E|(m + a) && (c) \\ = {} & |N|(m + a) + \sum_{b \in N \setminus I} |R_b|(m + a) + |N \setminus I|m \\ & + 2|O|a + |E|(m + a) \\ \leq {} & |N|(3m + 4a) + |E|(m + a) \end{aligned}$$

mit einem generellen Aufwand von $|E| + |N|$. Deswegen wird in der Regel in Simulatoren der *Back*propagation-Algorithmus eingesetzt. Seine Alternative findet trotzdem auch sinnvolle Anwendung, z. B. bei Real time recurrent learning (1.6.2). □

1.4 Kopplung von Gewichten

Die Methode des Gradientenabstiegs versucht, die Abbildung $\mathrm{Err}_T(\bullet)$ im $|E|$-dimensionalen Raum der Gewichte zu minimieren. Manchmal ist es sinnvoll, die Gewichte nicht als unabhängige Variablen zu betrachten, sondern aneinander zu koppeln; dies mag gewünscht sein, um Symmetrien oder Antisymmetrien in der Abbildung $p \mapsto \mathrm{out}_w(p)$ zu erreichen, oder um identische Teilnetze zu kodieren.

Beispiel Feature map (1.4.1) Eine *Feature map* ist ein neuronales Netz, bei dem sowohl die Eingabe- als auch die Ausgabeschicht mit einer Nachbarschaftsbeziehung ausgestattet ist (z.B. Anordnung der Neuronen in einer Fläche), wobei eine Erregung benachbarter Punkte in der Eingabeschicht zu einer Erregung benachbarter Punkte in der Ausgabeschicht führt. Kurzum, eine Feature map ist eine nachbarschaftserhaltende Abbildung von Erregungen durch ein neuronales Netz. Im menschlichen Gehirn bildet die sensorische Karte ein Analogon zur Feature map: Benachbarte Erregungen auf der Haut werden in benachbarten Gehirngebieten verarbeitet; hier interessiert, wie solche Karten durch Selbstorganisation entstehen.

Aber auch die explizite Konstruktion von Feature maps kann sich als nützlich erweisen. So wurden durch gekoppelte Gewichte aufgebaute Feature maps bei der Ziffernerkennung gewinnbringend eingesetzt [[le Cun 1989b]]. Dabei wird durch einen Knoten mit $n \times n$ Eingabeknoten ein sog. *Featuredetektor* aufgebaut. Er soll eine bestimmte Eigenschaft in seinem *rezeptiven Feld* (in der Anordnung seiner Eingabeknoten) erkennen, z. B. eine diagonale Linie o. ä. Diese Featuredetektoren werden kopiert, wobei entsprechende Gewichte gemeinsame Werte erhalten. Dadurch berechnen sie das jeweils gleiche Merkmal. Überlappen sich nun die rezeptiven Felder benachbarter Ausgabeknoten der Featuredetektoren, so entsteht eine Feature map (siehe Abb. 11). □

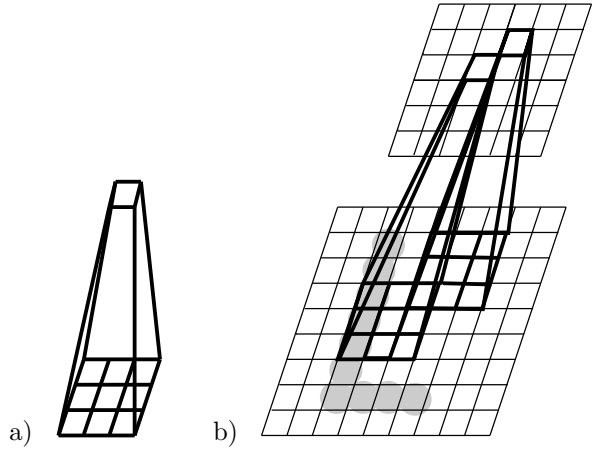


Abb. 11 a) Featuredetektor und b) daraus aufgebaute Feature map

Beispiel Antisymmetrie (1.4.2) Eine Standardtechnik, Computern Spiele beizubringen, ist es, zunächst in einer Spielsituation alle Handlungsmöglichkeiten aufzulisten und danach jede einzelne davon zu bewerten. Der Lösungsweg des ersten Teils für den sogenannten Zuggenerator ist meist durch die Spielregeln schon vorgezeichnet. Bei vielen Spielen (Dame, Schach, Backgammon, Abalone etc.) resultiert eine überschaubare, endliche Anzahl von Handlungsmöglichkeiten, die in einer geeigneten Weise als Spielpositionen kodiert werden können.

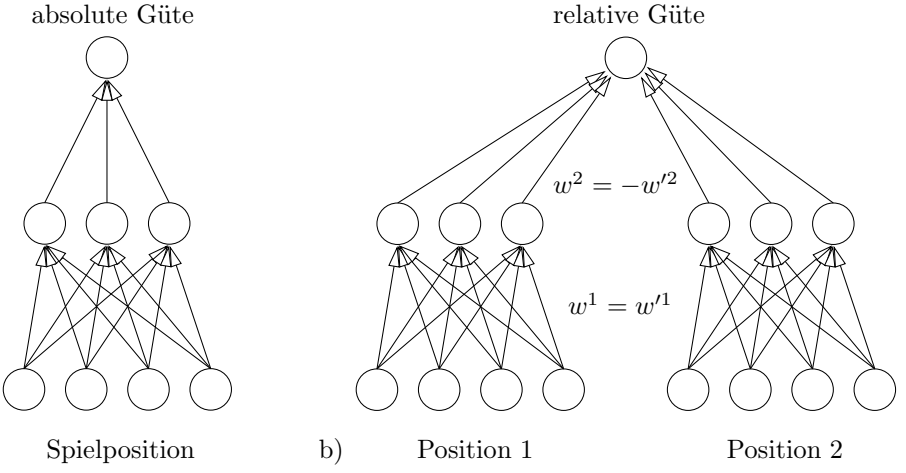


Abb. 12 Netz mit a) absoluter und b) relativer Güte von Spielpositionen

Es liegt nun nahe, einem neuronalen Netz Spielpositionen zu präsentieren. Ein Experte könnte diese z. B. mit einer Gütezahl im Intervall $[0, 1]$ bewerten (0 bedeute furchtbar und 1 exzellent) und ein neuronales Netz könnte diese Gütezahlen lernen (siehe Abb. 12a). Es zeigt sich jedoch in der Praxis, daß Experten oft Schwierigkeiten damit haben, konsistent eine absolute Güte von Positionen zu nennen. Leichter fällt es ihnen, diese zu vergleichen und eine relative Güte anzugeben, z. B. 0,1 für Position 2 ist deutlich besser als Position 1 und 0,9 für den umgekehrten Fall.

Diese Situation kann mit einem Netz mit gekoppelten Gewichten realisiert werden (siehe Abb. 12b): Ein Netz, das geeignet wäre, die absolute Güte von Spielpositionen zu lernen, wird kopiert. Entsprechende Gewichte im

kopierten und ursprünglichen Netz, die zwischen der Eingabe- und Binnenschicht vermitteln, haben gemeinsame Werte. Die entsprechenden Gewichte zwischen Binnen- und Ausgabeschicht werden so gekoppelt, daß sie gleichen Betrag, aber unterschiedliches Vorzeichen tragen.

Dadurch wird z. B. schon bewirkt, daß beim Vertauschen der Spielpositionen automatisch die relative Güte von x zu $1 - x$ wird (bei Verwendung der Fermifunktion am Ausgabeknoten). Das beschriebene Verfahren wurde erfolgreich für das Lernen von Backgammon angewendet ⟦Tesauro 1989⟧. □

Transformation d. Gewichte (1.4.3) Die Kopplung von Gewichten kann durch eine Nebenbedingung $h(w) = 0$ in Form einer Abbildung $h\colon \mathbb{R}^E \to \mathbb{R}$ formuliert sein oder z. B. dadurch, daß die Gewichte als differenzierbare Abbildung $w\colon \mathbb{R}^F \to \mathbb{R}^E$, $u \mapsto w(u)$, eines freien Variablensatzes $u \in \mathbb{R}^F$ mit $|F| \leq |E|$ angesehen werden. Es kann dann ein Gradientenabstieg

$$u^i = u^{i-1} - \eta \nabla \operatorname{Err}_T(w(\bullet)) \Big|_{u^{i-1}}$$

bezüglich der Variable u statt w in dem u. U. kleineren Raum $\mathbb{R}^F$ durchgeführt werden. Der Gradient einer reellwertigen Funktion stimmt bis auf Transponierung mit der Jacobimatrix überein (der Gradient ist ein Spalten- und die Jacobimatrix ein Zeilenvektor). Nach der Kettenregel ergibt sich ganz einfach

$$\mathrm{D} \operatorname{Err}_T(w(\bullet)) \Big|_{u^{i-1}} = \mathrm{D} \operatorname{Err}_T(\bullet) \Big|_{w(u^{i-1})} \cdot \mathrm{D}\, w \Big|_{u^{i-1}}.$$

Der Backpropagation-Algorithmus muß also nur dahingehend abgewandelt werden, daß zusätzlicher Speicherplatz für die Variable $u \in \mathbb{R}^F$ zur Verfügung gestellt wird, anstelle w die Variable u initialisiert wird, zu Beginn des Schritts (ii) w aus u berechnet wird und im Schritt (iii) statt $w \leftarrow w - \eta v$ nun

$$u \leftarrow u - \eta (\mathrm{D}\, w|_u)^T v$$

gebildet wird.

Der wesentliche Unterschied besteht also in der zusätzlichen Linksmultiplikation der von Anfang an bekannten und oft leicht zu berechnenden transponierten Jacobimatrix (siehe Abb. 13). Diese Matrix ist sogar konstant,

wenn zwischen u und $w(u)$ ein affiner (d. h. bis auf Addition einer Konstanten linearer) Zusammenhang besteht. Die Matrixmultiplikation erfordert i. allg. $|E||F|$ Multiplikationen und $(|E|-1)|F|$ Additionen. In vielen Fällen ist die Transformation $u \mapsto w(u)$ jedoch sehr einfach und der zusätzliche Aufwand auf einige wenige Operationen beschränkt, wie folgende Beispiele zeigen. □

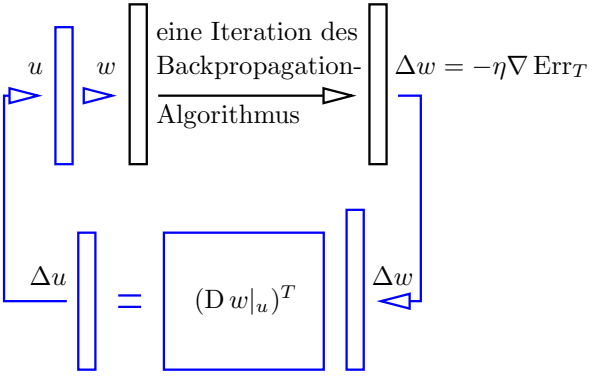


Abb. 13 Transformation für gekoppelte Gewichte

Eingefrorene Gewichte (1.4.4) Manchmal setzt sich ein größeres Netz aus kleineren Teilnetzen zusammen, die zum Teil schon vortrainiert sind. Deren Gewichte sollen dann nicht erneut gelernt werden. Ist $F \subset E$ die Indexmenge der noch zu trainierenden Teilmenge der Gewichte, so kann die Abbildung $w\colon \mathbb{R}^F \to \mathbb{R}^E$ aufgefaßt werden als

$$u \mapsto \left(e \mapsto \begin{cases} u_e & \text{falls } e \in F \\ \text{Konstante} & \text{sonst} \end{cases} \right).$$

Die Linksmultiplikation mit $(\mathrm{D}\, w)^T$ projiziert den Vektor v auf die noch zu trainierenden Komponenten (d. h. kopiert nur die zu trainierenden Komponenten und läßt die anderen weg). □

Gemeinsame Gewichte (1.4.5) Sollen Gewichte zweier Kanten e_1 und e_2 gleiche Werte haben (z. B. aufgrund von Symmetrien), so kann das z. B. durch folgende Abbildung $w\colon \mathbb{R}^{E\setminus\{e_2\}} \to \mathbb{R}^E$ ausgedrückt werden:

$$u \mapsto \left(e \mapsto \begin{cases} u_e & \text{falls } e \in E \setminus \{e_1, e_2\} \\ u_{e_1}/\sqrt{2} & \text{sonst} \end{cases} \right)$$

Die Matrixmultiplikation mit $(\mathrm{D}\, w)^T$, die Veränderung des Gewichtungssatzes u und die anschließende Neuberechnung von w aus u bewirkt, daß sowohl die e_1- als auch die e_2-Komponente von w den Mittelwert der Änderungen $-\eta \nabla \operatorname{Err}_T|_{w(u)}(e_1)$ und $\nabla \operatorname{Err}_T|_{w(u)}(e_2)$ für die (ungekoppelten) Richtungen e_1 und e_2 erfahren.

Es ist jedoch zu beachten, daß es bei der Festlegung der Transformation w viele Möglichkeiten gibt, gleiche Gewichtungskomponenten zu erreichen. Statt des Faktors $1/\sqrt{2}$ hätte ein beliebiger anderer Faktor ungleich null stehen können. Diese Festlegung kann im Einzelfall sowohl das Konvergenzverhalten als auch die Konvergenzgeschwindigkeit beeinflussen.

Mehrere gleiche Gewichte können in analoger Weise behandelt werden. □

Individuelle Lernraten (1.4.6) Gewichte haben einen typischen Arbeitsbereich, in dem sie etwas bewirken. Bei Verwendung der Fermifunktion treten z. B. bei Argumenten außerhalb eines Bereichs von etwa $[-5, 5]$ kaum Änderungen der Aktivation auf; der typische Arbeitsbereich von Gewichten liegt auch in diesem Intervall. Andere Aktivierungsfunktionen mögen andere sinnvolle Arbeitsbereiche der Gewichte nach sich ziehen. Um diesen u. U. verschiedenen Skalierungen Rechnung zu tragen, kann es sinnvoll sein, individuelle (d. h. in jede Koordinatenrichtung verschiedene, aber konstante) Lernraten zuzulassen. Das kann mit einer Transformation $w\colon \mathbb{R}^E \to \mathbb{R}^E$

$$u \mapsto (e \mapsto \sqrt{\eta_e} u_e)$$

mit $\eta_e \in \mathbb{R}^+$ bewerkstelligt werden. Sowohl die Anwendung der Transformation w als auch die Matrixmultiplikation mit $(\mathrm{D}\, w)^T$ entspricht einer komponentenweisen Multiplikation mit $\sqrt{\eta_e}$. Daher werden die Gewichte w_e um das η_e-fache des Fehlergradienten in Richtung w_e geändert. Die globale Lernrate kann dann natürlich gleich eins gesetzt sein.

Eine andere Motivation für individuelle Lernraten resultiert aus der Struktur der Aktivation eines Knotens a eines Backpropagation-Netzes: Würde jedes Gewicht um den Faktor $(1 + \eta)$ vergrößert, so verändert sich das Argument der Aktivierungsfunktion um $\eta \sum_{i \in L_a} w_{ia} h_{ai}$, und diese Größe

skaliert mit $|L_a|$, dem sog. *Fan-in* (Zahl der Vorgängerknoten). Daher kann es sinnvoll sein, für η_e anzusetzen, daß

$$\eta_e \propto 1/|L_{\Pi_2(e)}| \quad \text{für alle } e \in E. \qquad \square$$

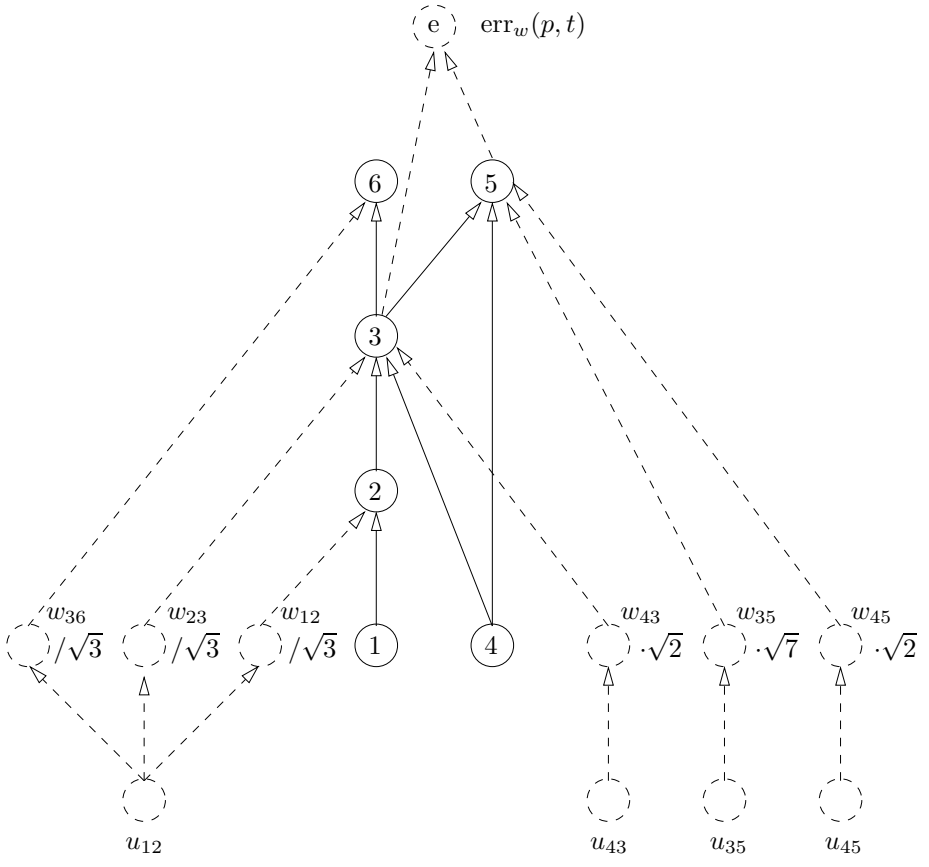


Abb. 14 Ein Abbildungsnetz mit gekoppelten Gewichten und individuellen Lernraten

Gewichtstransformation als Backpropagation-Netz (1.4.7) Meistens läßt sich die Transformation der Gewichte als Backpropagation-Netz mit konstanten Gewichten schreiben. Dann kann das im Beweis (1.3.2) verwendete Abbildungsnetz erweitert werden um das vorgeschaltete Netz der Transformation der Gewichte. Ein Beispiel ist in Abb. 14 gezeigt. Mit

der Delta-Regel des Theorems (1.3.1) finden sich dann sofort die richtigen Rechenregeln für die gekoppelten Gewichte. □

1.5 Design der Kostenfunktion

Die Ausgabeschicht eines Backpropagation-Netzes sollte Aktivierungsfunktionen haben, die in der Lage sind, die geforderten zu approximierenden Werte anzunehmen. Beispielsweise kann es schon bedenklich sein, mit der Fermifunktion $\sigma = x \mapsto 1/(1+\exp(-x))$ als Aktivierungsfunktion, binäre Werte approximieren zu wollen (siehe Abb. 15); das treibt die Gewichte in betragsmäßig große Werte, damit das Argument der Fermifunktion gegen $\pm\infty$ gehen kann. Zusätzlich ist problematisch, daß die Ableitung $\sigma' = \sigma \cdot (1-\sigma)$ von σ in der Nähe der „richtigen“ Stellen $\pm\infty$ verschwindend klein ist. Der an einem Ausgabeknoten n (ohne Nachfolgerknoten) zurückzupropagierende Fehler δ_n ist

$$\delta_n = \sigma'(o_n|_p) \cdot \partial d(t,y)/\partial y_n,$$

also proportional zu σ'. Schon deswegen ergeben sich kleine Gewichtungsänderungen, obwohl das Argument der Aktivierungsfunktion noch riesige Schritte machen müßte. $x \mapsto \frac{1}{2} + \tanh(x)$ als Aktivierungsfunktion vermeidet obige Probleme (siehe Abb. 15).

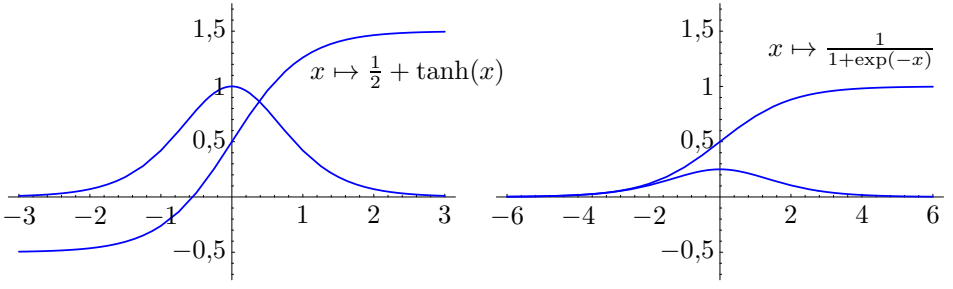


Abb. 15 $x \mapsto \frac{1}{2} + \tanh(x)$ und die Fermifunktion mit ihren Ableitungen

Andererseits kann es sein, daß die Komponenten der Netzfunktion die Bedeutung von Wahrscheinlichkeiten eines Bernoulli-Experiments (z. B. des Zutreffens einer Hypothese) haben sollen. Dann ist es außerordentlich erwünscht, daß die Komponenten der Netzfunktion Werte im Bereich $[0,1]$ haben; die Fermifunktion σ als Aktivierungsfunktion ist dann eine natürliche Wahl. Durch geschickte Wahl der Kostenfunktion kann die Proportionalität von δ_n zu $\sigma'(o_n|_p)$ eliminiert werden:

Relative Entropie (1.5.1) Die Wahl der *relativen Entropie*

$$\begin{aligned} d\colon (0,1)^O \times (0,1)^O &\to \mathbb{R} \\ (x,y) &\mapsto \sum_{n\in O} \left(x_n \log \frac{x_n}{y_n} + (1-x_n) \log \frac{1-x_n}{1-y_n} \right) \end{aligned}$$

als Kostenfunktion zusammen mit der Fermifunktion als Aktivierungsfunktion zieht einen an einem Ausgabeknoten n (ohne Nachfolgerknoten) zurückzupropagierenden Fehler

$$\delta_n = o_n|_p - t_n$$

– statt $\sigma'(o_n|_p) \cdot (o_n|_p - t_n)$ bei Verwendung der Standard-Kostenfunktion $(x,y) \mapsto (x-y)^2/2$ – nach sich. Die Kostenfunktion divergiert schon, wenn sich eine y-Komponente dem falschen Ende des Definitionsbereichs $(0,1)$ nähert. □

Freie Wahl der Ableitung (1.5.2) Im Backpropagation-Algorithmus wird die Ableitung $\partial d(t,y)/\partial y_n$ der Kostenfunktion benutzt, ihr Wert selbst jedoch nur, wenn der absolute Fehler in das Abbruchkriterium des Algorithmus einfließt. Im Prinzip könnte also die Ableitung in bestimmten Grenzen willkürlich festgelegt sein und damit der an der Ausgabeschicht zurückzupropagierende Fehler δ_n. Diese Freiheit kann benutzt werden, der Abbildung $w \mapsto \mathrm{Err}_T(w)$ ein für die Methode des Gradientenabstiegs günstiges Aussehen zu geben. In einer empirischen Untersuchung ⟦Fahlman 1989⟧ zeigte sich die Wahl der Kombination

$$\delta_n = \big(\sigma'(o_n|_p) + 1/10\big)\big(o_n|_p - t_n\big)$$

beiden in (1.5.1) diskutierten Fällen überlegen. In der gleichen Studie wird vorgeschlagen, die immer wieder in Ableitungen der Kostenfunktion auftauchende Komponente $o_n|_p - t_n$ der Differenz des berechneten mit dem gewünschten Vektor durch den Term

$$\operatorname{arctanh}\left(o_n|_p - t_n\right)$$

zu ersetzen (Wertebereich $(0,1)$ vorausgesetzt). arctanh, die Umkehrfunktion des tanh, divergiert für Argumente gegen ± 1, also genau dann wenn die berechnete Komponente am „falschen“ Ende des Wertebereichs liegt. Fehler werden hier – nichtlinear – verstärkt berücksichtigt. □

Strafterme in der Kostenfunktion (1.5.3) Die Kostenfunktion soll eigentlich die Abweichung zweier Vektoren des $\mathbb{R}^O$ messen. Im Prinzip könnte sie jedoch auch noch zusätzliche Terme enthalten, die eine unerwünschte Eigenschaft des Netzes, z. B. Gewichte hohen Betrags, bestrafen. Die ursprüngliche Kostenfunktion d_o könnte erweitert werden:

$$d(x,y) = d_o(x,y) + \kappa \sum_{e \in E} w_e^2, \quad \kappa \in \mathbb{R}^+$$

Das führt dann zu einem noch explizit von den Gewichten abhängigem Gesamtfehler. Die Methode des Gradientenabstiegs bewirkt dann, daß jedes Gewicht w_e pro Iteration des Backpropagation-Algorithmus die zusätzliche Änderung

$$w_e \leftarrow (1-\varepsilon) w_e$$

mit $\varepsilon = 2|T|\eta\kappa$ erfährt. Dies gibt auch schon einen Anhaltspunkt für die Wahl des neuen Parameters; κ muß so gewählt sein, daß die Gewichte nur eine moderate Verkleinerung ihres Betrags erfahren, ε also klein ($\ll 1$) ist. Durch diese Art der Bestrafung von Gewichten mit hohem Betrag wird unter anderem bewirkt, daß nicht benötigte Gewichte zu Null zerfallen. Das ist oft genau der gewünschte Effekt. Sollen Gewichte kleinen Betrags schneller zerfallen als solche großen Betrags (anstatt Gewichte großen Betrags unverhältnismäßig zu reduzieren), so könnte obige Änderung mit

einem von der Kante e abhängigem ε helfen, z. B. ⟦vgl. Hanson und Pratt 1989⟧

$$\varepsilon_e = \frac{2|T|\eta\kappa}{\left(1 + w_e^2\right)^2}.$$ □

Forward models (1.5.4) Manchmal ist es nicht von Interesse, ein Netz mit einer bestimmten Funktion f zu trainieren, sondern gerade eine Umkehrung einer Funktion soll gelernt werden. Sei beispielsweise ein Roboterarm mit drei Gelenken gegeben, der Punkte in der Ebene erreichen kann. Die *Kinematik* des Roboterarms, d. h. diejenige Abbildung f, die den Winkeln einen Ort zuordnet, ist einfach durch die Geometrie des Arms gegeben. Will man aber einen bestimmten Punkt der Ebene ansteuern, so muß man zugehörige Winkeltripel kennen, von denen es mehrere geben mag (f ist nicht injektiv).

Im allgemeinen sei $f\colon \mathbb{R}^O \to \mathbb{R}^I$ eine nicht notwendigerweise injektive Abbildung. Ein neuronales Netz soll so konstruiert werden, daß bei Eingabe eines p irgendein $t \in f^{-1}(\{p\})$, also ein t mit $f(t) = p$, annähernd vom Netz ausgegeben wird.

Es ist naheliegend, zufällig ausgesuchte Trainingsmuster $(f(t), t)$ lernen zu lassen. Da die Umkehrung von f nicht eindeutig sein muß, kann es widersprüchliche Informationen im Datensatz geben, z. B. Werte (p, t_1) und (p, t_2), wobei $t_1 \neq t_2$. Backpropagation-Netze mit der Standard-Kostenfunktion des quadratischen Abstandes lernen dann den Mittelwert von t_1 und t_2 an der Stelle p. Es ist aber unmittelbar einleuchtend, daß f angewendet auf den Mittelwert nicht notwendigerweise p ergibt. Das trifft vor allem dann zu, wenn f in einer Weise nichtlinear ist, daß die Menge $f^{-1}(\{p\})$ nicht konvex ist: In einer *konvexen Menge* liegt die Verbindungslinie irgend zweier Punkte in der Menge. Bei einer nichtkonvexen Menge $f^{-1}(\{p\})$ finden sich also immer Trainingsdaten, bei denen die gemittelten Targets für ein bestimmtes p nicht zu $f^{-1}(\{p\})$ gehören (siehe Abb. 16).

Diese Methode, auch *direktes inverses Modell*, genannt ist in Abb. 17a dargestellt und führt i. allg. höchstens für injektive f zu einem befriedigenden Resultat. Oft ist es zu teuer, einen Datensatz mit sinnvollen $(f(t), t)$-Paaren zu generieren; es reicht nämlich nicht aus, widersprüchliche Infor-

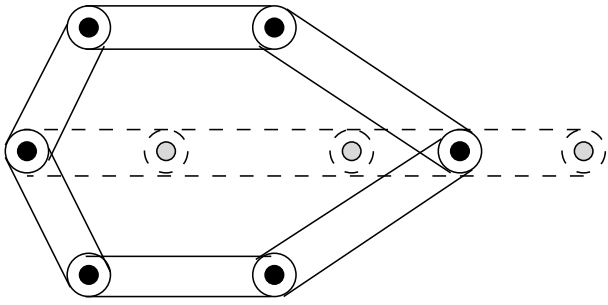


Abb. 16 Zwei verschiedene Winkeleinstellungen eines Roboterarms führen zum gleichen Ort. Die Einstellung des Mittelwerts der Winkel (gestrichelt) bewirkt jedoch einen anderen Zielort.

mationen auszumerzen. Vielmehr soll ein stetiger Zweig von f^{-1} gefunden werden. Es ist ja beispielsweise nicht erwünscht, daß der Roboterarm zum Ansteuern zweier benachbarter Punkte völlig verschiedene Winkeleinstellungen bekommt, selbst wenn diese zum richtigen Ort führen.

Eine andere Möglichkeit ist es, ein Teilnetz so vorzutrainieren, daß es f approximiert. Dieses sog. *Forward model* kann mit eingefrorenen Gewichten dem zu trainierenden Netz nachgeschaltet werden; als Trainingsrelation wird die Identität auf $\mathbb{R}^I$ verwendet (siehe Abb. 17b) mit der quadratischen Abweichung als Kostenfunktion. So wird das zu trainierende Teilnetz dazu gezwungen, Werte zu liefern, die durch das Modell von f auf die Identität abgebildet werden; mithin lernt es eine Inverse von f. Dieser Ansatz bringt sehr gute Resultate, auch wenn f nicht perfekt durch das Forward model wiedergegeben wurde ⟦Jordan und Rumelhart 1990⟧.

Angenommen f ist differenzierbar. Dann ist die Benutzung eines perfekten Forward models äquivalent dazu, das gleiche Netz wie beim direkten inversen Modell zu verwenden, jedoch statt des Standard-Einzelfehlers den folgenden Ausdruck

$$\mathrm{err}_w(p,t) = (t - f(\mathrm{out}_w(p)))^2/2$$

zu benutzen und die Identität zu lernen, d. h. $t = p$. Dies bedeutet die Verwendung der „Kostenfunktion“ d, definiert durch $(t, y) \mapsto (t - f(y))^2/2$. Diese Abbildung paßt formal nicht in das vorgegebene Konzept der Eigenschaften einer Kostenfunktion, z. B. lebt sie nicht auf $\mathbb{R}^O \times \mathbb{R}^O$, sondern auf

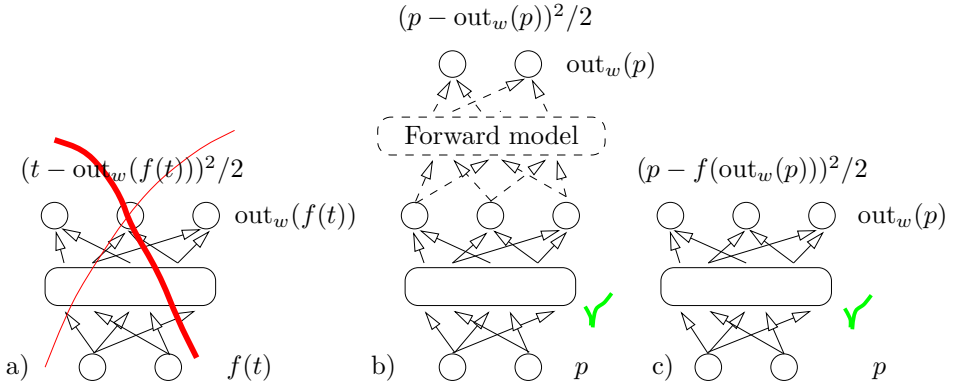


Abb. 17 Kostenfunktionen zum Lernen einer inversen Funktion

$\mathbb{R}^I \times \mathbb{R}^I$, weil die Netzausgabe vermöge f nachbearbeitet wurde. Dies soll aber nicht weiter stören, denn der Backpropagation-Algorithmus kann unverändert angewendet werden. Nach wie vor bekommt δ_n am Knoten n der Ausgabeschicht den Anteil $\partial(y \mapsto d(t,y))/\partial y_n$, der durch den Einzelfehler herrührt. Es ergibt sich hier speziell

$$\frac{\partial(y \mapsto d(t,y))}{\partial y_n} = \big((f(y)-t)^T \,\mathrm{D}\, f\big)_n = \sum_{i\in I}(f_i(y)-t_i)\frac{\partial f_i(y)}{y_n}.$$

Manchmal sind Simulatoren nicht so flexibel, eine solche Formel zu benutzen; auch ist f u. U. nicht differenzierbar, nur nach Beispielen bekannt oder mit zufälligen Störungen versehen. In diesen Fällen ist dann die Methode des Forward models angebracht, ansonsten die der speziellen „Kostenfunktion“. □

1.6 Anwendung auf rekurrente Backpropagation-Netze

Sind allgemeine endliche gerichtete Graphen als Grundlage eines neuronalen Netzes gestattet, so können Rückkopplungen auftreten. Es ist dann naheliegend, nach der *Dynamik* eines solchen Netzes zu fragen. Zum Beispiel mag gewünscht sein, daß die Funktionswerte synchron an den einzelnen Knoten zu diskreten Zeitschritten berechnet werden. Hierbei entstehen an den Knoten Folgen o_n^1, o_n^2, … von Aktivationen (zu den Zeitpunkten

1, 2, ...). Generell ist es bei rekurrenten Netzen wünschenswert, auch zu beliebigen Zeitpunkten i Eingaben p_a^i an einem Knoten a vornehmen zu können. Ziel kann es jetzt sein, dem Netz beizubringen, wie es eingegebene Folgen erkennt, bestimmte Folgen ausgibt oder ergänzt (*Autoassoziation*) oder im allgemeinen Fall eine bestimmte Folge ausgibt, wenn eine andere Folge eingegeben wurde (*Heteroassoziation*).

Backpropagation through time (1.6.1) Die Dynamik eines Netzes sei

$$o_b^{i+1} = f_b\Big(\sum_{a\in L_b} w_{ab} o_a^i + p_b^i\Big);$$

das bedeutet synchrone Updates und diskrete Zeitschritte. Liegt zur Zeit i keine Eingabe an, dann ist einfach $p_b^i = 0$ zu setzen. Soll ein solches Netz nun eine Abfolge von Aktivationen für beschränkt viele, z. B. maximal L, Zeitpunkte liefern, dann kann es in ein Feedforward-Netz umgewandelt werden, das mit dem Backpropagation-Algorithmus arbeitet (siehe Abb. 18): Dazu werden die Knoten L-mal kopiert, wobei die Kopien durch fortlaufende Indizes unterschieden werden. Eine Kante (a, b) des ursprünglichen Netzes wird $L-1$-mal in das neue Netz als (a^1, b^2), ..., (a^{L-1}, b^L) eingefügt; die entsprechenden Gewichte werden miteinander identifiziert $w_{ab} = w_{a^1b^2} = \ldots = w_{a^{L-1}b^L}$.

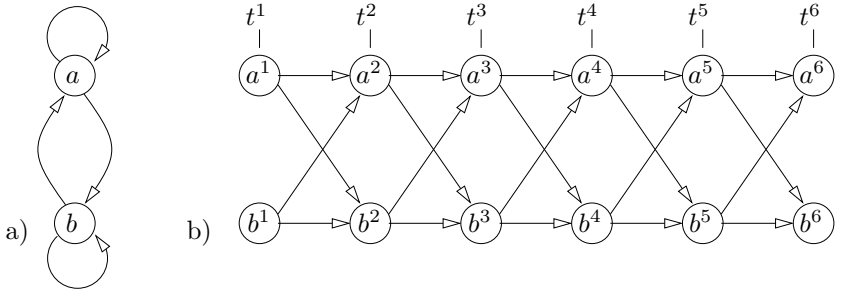


Abb. 18 Backpropagation through time: Ein rekurrentes Netz a) wird in ein Feedforward-Netz b) umgewandelt, an dem Zeitsequenzen (z.B. Noten $t^1, t^2, \ldots$ eines Musikstücks) als Zielmuster angelegt werden können.

Beliebige Knoten des aufgefalteteten Netzes können zusätzlich eine Eingabe bekommen, Ausgabeknoten sein, nichts davon oder beides. Dem entspricht eine Vorgabe an dem einen oder anderen Knoten zu bestimmten Zeitpunkten als Eingabe oder Target. Das aufgefaltete Netz wird nun mit Backpropagation trainiert: Zu identifizierende Gewichte würden dabei während einer Iteration zunächst verschiedene Änderungen erfahren. Aus dem Unterabschnitt (1.4.5) über Weight sharing geht hervor, daß die Summe der einzelnen Änderungen als gemeinsame Änderung verwendet werden muß. So bleibt die Kopplung der Gewichte erhalten. □

Rekurrentes Lernen (1.6.2) Eine andere Methode kommt ohne Duplizieren der Knoten aus. Hier wird die gleiche Dynamik wie bei Backpropagation through time angenommen. Zu jedem Zeitpunkt i kann die Netzausgabe an einer bestimmten Teilmenge $O^i \subset N$ mit einer Kostenfunktion $d^i(\bullet,\bullet)$ und gewünschtem Ausgabenvektor $t^i \in \mathbb{R}^{O^i}$ bewertet werden.

Anstatt jetzt das Netz soweit aufzufalten, wie nötig ist, um einen nach vielen Zeitschritten erhaltenen Fehler zurückzupropagieren, kann in jedem Zeitschritt mit Hilfe der Backpropagation-Alternative (1.3.4) eine Ableitung der Kostenfunktion d^i berechnet werden zu

$$\Delta w_{ab}^i = -\eta \sum_{n \in O^i} \partial d^i(t^i, y)/\partial y_n \Big|_{o^i(p^i)} \gamma^{i\,n}_{ab}$$

mit der bekannten Rekursion der Gammas

$$\gamma^{i+1\,n}_{ab} = f_n'(h_n^i)\Big(\delta_{nb} o_a^i + \sum_{m \in L_n} \gamma^{i\,m}_{ab} w_{mn}^i\Big).$$

Hierbei ist δ_{nb} das sog. *Kronecker-Symbol*, das genau dann 1 ist, wenn $n = b$ und 0 sonst. Für die Randbedingung der Gammas zum Zeitpunkt $i = 0$ wird sinnvollerweise Null angesetzt: $\gamma^{o\,n}_{ab} = 0$.

Bei dieser Sorte Netz kann die günstige Induktionsrichtung der Gammas dazu ausgenutzt werden, nicht nur die Aktivationen zu propagieren, sondern auch gleichzeitig das *Credit assignment* Δw^i für diesen Zeitschritt zu berechnen. So können die aufsummierten Änderungen $\Delta w_{ab} = \sum_i \Delta w_{ab}^i$

am Ende eines gesamten Intervalls auf den Gewichtungsvektor angewendet werden. Oder aber die Änderung $\Delta w_\bullet^i$ wird sofort nach jedem Zeitschritt auf den Gewichtungsvektor w angewendet; dies ist auch als *Real time recurrent learning* ⟦Williams und Zipser 1989⟧ bekannt. □

1.7 Wertung

Das Anliegen dieses Kapitels war es, die notwendige mathematische Grundlage zum Studium von Backpropagation-Netzen in einheitlicher und möglichst allgemeiner Darstellung zu geben. Die Ableitung vieler verschiedener Lernregeln wird in der Lehrbuchliteratur immer wieder als Spezialfall gegeben, anstatt den zugrundeliegenden Mechanismus aufzudecken. Die vorgestellten Kettenregeln für allgemeine Abbildungsnetze sind so nützliche Werkzeuge, daß sie nicht nur bei der Ableitung des Backpropagation-Algorithmus oder beim Aufstellen von Lernregeln in rekurrenten Netzen eine tragende Rolle spielen, sondern z. B. auch später bei der exakten Berechnung von Wechselwirkungen in Boltzmann-Maschinen (6.2.14).

Die Analyse zeigt z. B., daß der Backpropagation-Algorithmus in seiner effizienten Methode besticht, den Gradienten einer Fehlerfunktion zu berechnen. Dies kostet nämlich genauso viel Zeit wie die Bestimmung nur einer einzigen Komponente des Gradienten bei einer (ungenauen) numerischen Differentiation $(\mathrm{err}_{w'}(p,t) - \mathrm{err}_w(p,t))/|w' - w|$.

Weitere zeitliche Vorteile ergeben sich aus einer einfachen Kostenfunktion (z. B. der quadratischen Abweichung oder der relativen Entropie) oder der Verwendung spezieller Aktivierungsfunktionen wie der Fermifunktion: Hier läßt sich die oft benötigte Ableitung $f_a'(r) = f_a(r)(1 - f_a(r))$ aus einfachen Operationen aus dem sowieso benötigten Aktivationswert $f_a(r)$ bestimmen. Das mag bei der Erstellung vieler Simulatoren der Grund dafür gewesen sein, sich auf solche Modelle zu beschränken.

Andere Einschränkungen bestehender Simulatoren sind schon schwerwiegender. Daß oft keine frei Wahl bei der Ausgabeschicht gewährt wird, mag daran liegen, daß in Lehrbüchern nicht auf diese Situation eingegangen wird. Bei freier Wahl der Ausgabeschicht können an einem Knoten nämlich zwei Fehlerursachen identifiziert werden: die Abweichung aufgrund einer

gewünschten Netzausgabe an diesem Knoten und zurückpropagierte δ-Fehleranteile aufgrund von Abweichungen an anderen Ausgabeknoten in höheren Schichten. Das Backpropagation-Theorem (1.3.1) in diesem Kapitel zeigt, daß beide Fehleranteile einfach zu addieren sind.

Die Berechnungen in Backpropagation-Netzen sind alle lokal, d. h. benötigen nur lokale, zu einzelnen Knoten assoziierte Größen. Die Berechnung des Gradienten ist jedoch *nicht zeitlich lokal*: Für die Gewichtungskomponente w_{ab} wird die Aktivationen vom Knoten b bis hin zur letzten Ausgabe in topologischer Reihenfolge propagiert und dann die aufgetretenen Fehler wieder in entgegengesetzter Richtung bis zum Knoten b zurückpropagiert. Besonders augenfällig ist das bei rekurrenten Netzen, die in der Zeit aufgefaltet werden.

Der natürliche Geschwisteralgorithmus von Backpropagation aus dem Unterabschnitt (1.3.4), ist hingegen in diesem Sinn zeitlich lokal, da die Induktionsrichtung für die Berechnung des Gradienten die gleiche ist wie für die Berechnung der Aktivationen. Der größere Rechen- und Speicheraufwand kann sich bei rekurrenten Netzen auszahlen, die sofort während des Propagierens schon den Gewichtungssatz ändern wollen (Real time recurrent learning).

Die Bewertung des Backpropagation-Algorithmus als Algorithmus zur Minimierung der Fehlerfunktion, d. h. die Bewertung der Methode des Gradientenabstiegs, geschieht im nächsten Kapitel.

2 Modifikationen von Backpropagation

Simply modify the sigmoid-prime function so that it does not go to zero for any input value.

— S. E. Fahlman ⟦Fahlman 1989⟧

Das Basisproblem des vorherigen Kapitels war es, ein Minimum einer differenzierbaren Funktion $\mathrm{Err}_T : \mathbb{R}^E \to \mathbb{R}$ zu finden. Die Methode des Gradientenabstiegs ist ein lokales Verfahren, d. h. findet ein lokales Minimum im Einzugsbereich der Startbesetzung der Gewichte. Das mindeste, was von einem Verfahren zur Anpassung des Gewichtungssatzes erwartet wird, ist daß es bei Startwerten in der Nähe eines typischen Minimums dieses in effektiver Weise findet.

Das Trade-off-Theorem für Backpropagation (2.1.2) besagt i. w., daß immer bestimmte Minima übersehen werden. Darüberhinaus ist es nicht sehr effizient. In seiner Doktorarbeit hat R. Salomon ein Verfahren entwickelt, das den ersten Nachteil nicht hat ⟦Salomon 1991⟧ und in einer vergleichenden Studie ganz gut abschneidet ⟦Moreira und Fiesler 1995⟧.

Es zeigt sich jedoch, daß dieses Verfahren keineswegs asymptotisch stabil in einem noch zu definierenden Sinn (2.2.2;v) ist. Ziel dieses Kapitels ist es, ein kompetitives und stabiles Lernverfahren zu entwickeln, das effizient für die Boltzmann-Maschine in Abschnitt 6.5 eingesetzt werden kann.

2.1 Trade-off-Theorem für Backpropagation

Reguläres Minimum (2.1.1) Ein *reguläres Minimum* w^* einer in eine Taylorreihe entwickelbaren Fehlerfunktion ist dadurch gekennzeichnet, daß die erste Ableitung $\mathrm{D}\,\mathrm{Err}_T$ an der Stelle w^* null ist, und die zweite Ableitung $\mathrm{D}^{(2)}\,\mathrm{Err}_T = \mathrm{D}\,\mathrm{D}\,\mathrm{Err}_T$ hier positiv definit ist, d. h. die *quadratische Form* $w \mapsto \mathrm{D}^{(2)}\,\mathrm{Err}_T\,|_{w^*}(w)(w)$ für von Null verschiedene Argumente positiv ist. Für Gewichtungssätze nahe genug an w^* können zur ausreichenden Approximation die höheren Ableitungen der Taylorreihe

weggelassen werden. Statt in Err_T kann dann in der Abbildung $w \mapsto \mathrm{D}^{(2)}\,\mathrm{Err}_T\,|_{w^*}(w-w^*)(w-w^*)/2$ ein Minimum gesucht werden (auch das Weglassen des konstanten Terms $\mathrm{Err}_T(w^*)$ ändert ja nichts an der Position des Minimums). Es ist gebräuchlich, die zweite Ableitung einer reellwertigen Funktion als sog. *Hessesche Matrix* H zu schreiben, deren ij-te Komponente die partielle Ableitung $\partial^2\,\mathrm{Err}_T\,/\partial w_i \partial w_j$ ist. Zur weiteren Vereinfachung wird $w^* = 0$ angenommen. Dann lautet die quadratische Form in Matrixschreibweise $w \mapsto w^T H|_o w/2$ (siehe Abb. 19).

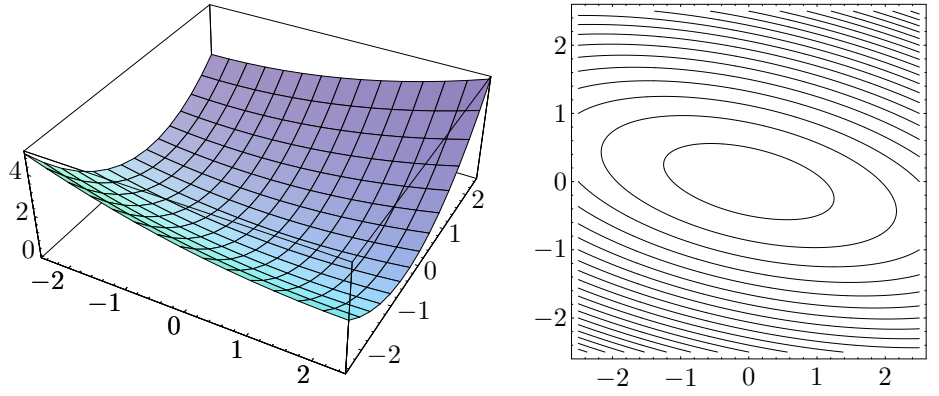


Abb. 19 Reguläres Minimum in 3D-Darstellung oder als 2D-Höhenlinienbild

Seien λ_e mit $e \in E$ die *Eigenwerte* der symmetrischen Matrix $H|_o$ (d. h. $\lambda_e\,\mathrm{I}\,w = H|_o w$ für ein $w \neq 0$); dann kann das Koordinatensystem so gedreht werden (*Hauptachsentransformation*), daß im gedrehten Koordinatensystem die Minimumsuche der Fehlerfunktion auf die Minimumsuche der diagonalen quadratischen Form $Q\colon \mathbb{R}^E \mapsto \mathbb{R}$ mit

$$Q(w) = \sum_{e \in E} \lambda_e w_e^2/2$$

zurückgeführt werden kann. Diese Abbildung repräsentiert mit Parametern $\lambda_e \in \mathbb{R}^+$, $e \in E$, immer noch den allgemeinen Fall eines typischen Minimums in der Fehlerfunktion. Die Frage lautet, wie gut oder schnell von einem Lernverfahren das Minimum 0 dieser Abbildung gefunden wird. Das klärt das *asymptotische Verhalten* des Verfahrens.

Es ist jedoch dabei zu beachten, daß das untersuchte Verfahren keinen expliziten Bezug auf das Koordinatensystem nimmt (z.B. eindimensionale Minimierungsschritte in Koordinatenachsenrichtung, individuelle Lernrate je nach Koordinatenrichtung ...). Denn solche Verfahren können unfaire Vorteile aus obiger Drehung des Koordinatensystems ziehen. Der Backpropagation-Algorithmus kann in Termen des Gradienten koordinatensystemfrei formuliert werden, und die Drehung des Koordinatensystems vereinfacht demzufolge nur die Analyse des Algorithmus, nicht aber das Minimierungsproblem. □

Trade-off-Theorem (2.1.2) *Sei $E: \mathbb{R}^n \to \mathbb{R}$ eine Fehlerfunktion eines Backpropagation-Netzes mit Minimum $w^* \in \mathbb{R}^n$. E sei um w^* in eine Taylorreihe entwickelbar und die Hessesche Matrix $H|_{w^*}$ an der Stelle w^* sei positiv definit mit größtem Eigenwert λ. Der Backpropagation-Algorithmus mit Lernrate $\eta > 2/\lambda$ kann im allgemeinen nicht gegen w^* konvergieren.* □

Beweis (2.1.3) Für die Konvergenzbetrachtung reicht es, die Taylorreihe um w^* nach dem dritten Glied abzubrechen,

$$E(w) = E(w^*) + \nabla E \Big|_{w^*}^T (w - w^*) + \frac{1}{2}(w - w^*)^T H \Big|_{w^*} (w - w^*) + \dots$$

Wegen der Minimumeigenschaft von w^* verschwindet $\nabla E|_{w^*}$. Sei nun U eine orthogonale Matrix, die $H|_{w^*}$ diagonalisiert, d. h. $\Lambda := U^T H|_{w^*} U$ ist diagonal. Da $H|_{w^*}$ symmetrisch ist, gibt es so ein U immer: die Spaltenvektoren von U sind die Hauptachsen, die Diagonalelemente von Λ die Eigenwerte λ_e der Matrix $H|_{w^*}$. Sei $x := U^T(w - w^*)$, dann reicht es zu zeigen, daß der Backpropagation-Algorithmus für bestimmte Startvektoren x^o unter der Fehlerfunktion $Q: x \mapsto x^T \Lambda x/2$ nicht gegen 0 konvergiert. Wegen der Orthogonalität von U nimmt nämlich Q an der Stelle 0 sein Minimum an, wenn E an der Stelle w^* ein Minimum annimmt:

$$\begin{aligned} Q(x) &= \frac{1}{2} x^T \Lambda x \\ &= \frac{1}{2}(U^T(w - w^*))^T U^T H \Big|_{w^*} U U^T (w - w^*) \\ &= \frac{1}{2}(w - w^*)^T H \Big|_{w^*} (w - w^*) \\ &\approx E(w) - E(w^*) \end{aligned}$$

Die explizite Form der e-Komponente des Gradienten von Q an der Stelle x lautet $\lambda_e x_e$. Folglich wird bei jedem Schritt des Backpropagation-Algorithmus zur Minimierung von Q die e-Komponente des Gewichtungssatzes mit dem Faktor $(1 - \eta\lambda_e)$ multipliziert. Der k-te Gewichtungssatz x^k hat also die e-Komponente

$$x_e^k = x_e^o(1 - \eta\lambda_e)^k.$$

Diese muß für $\eta > 2/\lambda_e$ divergieren, falls $x_e^o \neq 0$. Sind alle Komponenten des Startvektors x^o verschieden von Null (das ist mit Wahrscheinlichkeit 1 der Fall bei normalverteiltem x^o), so kann für $\eta > 2/\lambda$ mit $\lambda = \max_e \lambda_e$ der Backpropagation-Algorithmus nicht gegen 0 konvergieren. ∎

Bemerkungen (2.1.4) (i) Die Voraussetzungen des obigen Satzes sind recht allgemein. Die wesentliche Lektion lautet: Die Methode des Gradientenabstiegs – und daher auch der Backpropagation-Algorithmus – übersieht Minima, deren Hessesche Matrix einen größten Eigenwert von mehr als $2/\eta$ aufweist. Das ist eine typische Trade-off-Situation: Eine große Lernrate bewirkt zwar große Schritte im Gewichtungsraum (mit der Hoffnung nach schneller Konvergenz), schränkt aber den Typ von Minimum ein, der potentiell gefunden werden kann.

(ii) Zu Beginn des Lernens sind schon nicht die Minima der Fehlerfunktion bekannt, geschweige denn die Eigenwerte der Hesseschen Matrix an den Stellen der Minima. Nun sind in neuronalen Netzen unterschiedliche Gewichte unterschiedlich wichtig: Das heißt, daß die Änderungen des Gradienten bezüglich verschiedener Richtungen im Gewichtungsraum deutlich unterschiedlich ausfallen. Somit sind große Unterschiede in den Eigenwerten der Hesseschen Matrix zu erwarten und in Anwendungen typisch. Eine obere Grenze für die Eigenwerte (und damit eine untere für die Lernrate) kann nicht angegeben werden.

(iii) Ein Ansteigen des Gesamtfehlers $t \mapsto \mathrm{Err}_T(w^t)$ im Verlauf des Trainings ist ein sicheres – und ein Stagnieren des Gesamtfehlers ein mögliches – Zeichen für eine zu große Lernrate. Unerfahrene Trainer von neuronalen Netzen neigen dazu, hier das Gefangensein in lokalen Minima zu sehen, obwohl das vielleicht einzige Minimum durch eine zu hohe Lernrate übersehen wurde. □

Reduzierung der Dimensionalität (2.1.5) Sind alle Eigenwerte der Hesseschen Matrix gleich, so zeigt der Gradient der quadratischen Form immer in die Richtung des Ortsvektors, der ja zugleich (bis auf das Vorzeichen) die Richtung zum Minimum 0 ist. Die Kugelsymmetrie dieses speziellen Falls könnte durch Transformation in ein anderes Koordinatensystem auf einen eindimensionalen Fall zurückgeführt werden.

Sind im allgemeinen Fall die Eigenwerte unterschiedlich, so entsteht bei der Anwendung der Methode des Gradientenabstiegs die Situation, daß die Richtung des Gradienten ∇Q an einer Stelle w nicht mit der Richtung zum Minimum übereinstimmen muß (siehe Abb. 20). Diese Richtungsunterschiede können umso größer sein, je unterschiedlicher die Eigenwerte der Hesseschen Matrix am Minimum ausfallen. Der ungünstigste Fall wird also durch den Quotienten $\lambda := \max_{e\in E}\{\lambda_e\}/\min_{e\in E}\{\lambda_e\}$ des größten zum kleinsten Eigenwert bestimmt. Die allgemeinen Schwierigkeiten des Suchens eines Minimums in der Abbildung $Q\colon \mathbb{R}^E \to \mathbb{R}$ mit dem Gradientenverfahren sind also auch schon in der Abbildung

$$w \mapsto w_1^2/2 + \lambda w_2^2/2 \quad \text{mit} \quad \lambda \in \mathbb{R}^+$$

von $\mathbb{R}^2$ nach $\mathbb{R}$ enthalten. Das ist eine ausreichende Motivation, verschiedenste Fehlerminimierungsalgorithmen anhand dieser Fehlerabbildung zu untersuchen. □

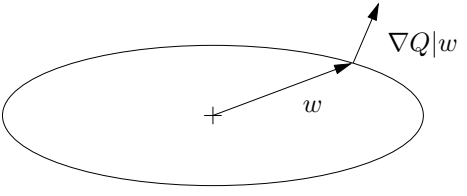


Abb. 20 Der Gradient am Punkt w zeigt nicht in Richtung Minimum

Konvergenz in Abhängigkeit der Lernrate (2.1.6) Die diskrete Methode des Gradientenabstiegs ergibt, beginnend bei einem Startpunkt w^o, die induktiv definierte Folge von Punkten

$$w^{i+1} = w^i - \eta(w^i{}_1, \lambda w^i{}_2).$$

Das Konvergenzverhalten dieser Folge hängt außerordentlich von der Lernrate η ab (siehe Abb. 21). Das Trade-off-Theorem zeigt, daß für Lernraten größer als einem kritischen Wert

$$\eta^c := 2/\lambda$$

die Folge w^o, w^1, ... nicht konvergiert, denn die zweiten Komponenten der Folgeglieder bilden dann eine alternierende Folge mit wachsendem Betrag (vorausgesetzt die zweite Komponente ${w^o}_2$ des Startwerts für die Gewichte war verschieden von Null).

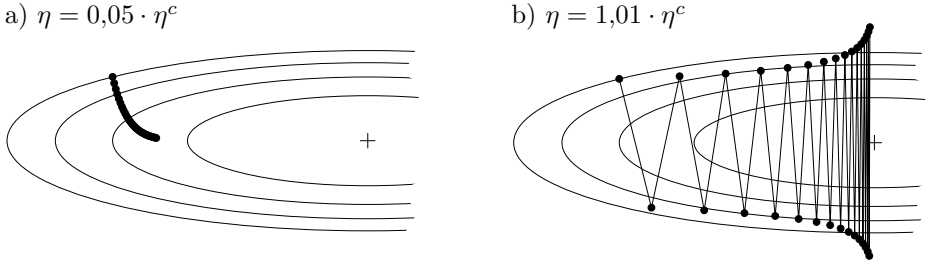


Abb. 21 Die Folge der Gewichtungssätze bei der Methode des Gradientenabstiegs für die Abbildung $w \mapsto w_1^2/2 + \lambda w_2^2/2$ mit dem Startpunkt $(-2, 2/\sqrt{\lambda})$ und $\lambda = 16$. a) Eine kleine Lernrate ergibt eine Folge von Gewichtungssätzen, die der Lösungskurve des kontinuierlichen Falls sehr nahe kommt (siehe Abb. 26). b) Schon mit einer Lernrate, die die kritische um 1% übersteigt, zeigt sich nach anfänglicher Verkleinerung des Fehlers schnell eine Divergenz der Folge der Gewichtungssätze.

Die erste Komponente wäre mit $\eta = 1$ nach dem ersten Schritt beim optimalen Wert 0 angelangt (bei $\lambda > 1$ würde aber die zweite Komponente jedoch nicht konvergieren). Für $\eta = 1/\lambda$ wäre die zweite Komponente in einem Schritt beim Minimum, dafür dauert es relativ lange, bis die erste Komponente in der Nähe von 0 angelangt ist. Irgendwo zwischen diesen Werten liegt die optimale Lernrate für diesen Schritt (abhängig vom Startpunkt w^o und λ). Eine gute Lernrate, z. B. $\eta = \lambda\eta^c/(\lambda + 1)$, liegt knapp unter der kritischen Lernrate η^c: typische Nähmaschinenschritte ergeben dann eine relativ schnelle Konvergenz bei fester Lernrate (siehe Abb. 22). □

Ein Versuch, die oft beobachteten Nähmaschinenschritte zu glätten, besteht darin, die vorherige Gewichtungsänderung abgeschwächt in die Änderungsformel der Gewichte einfließen zu lassen.

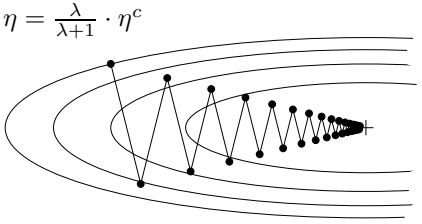


Abb. 22 Nähmaschinenschritte bei einer optimalen Lernrate für diesen Startwert

Gradientenabstieg mit Impuls (2.1.7) Eine Modifikation des Gradientenabstiegs in einem Backpropagation-Netz mit Fehlerfunktion E besteht in dem *Impulsverfahren*: Wähle $w^o \in \mathbb{R}^E$ zufällig oder beliebig. Setze $\Delta w^o = 0$ und für $i > 0$

$$\Delta w^i = -\eta \nabla E|_{w^{i-1}} + \alpha \, \Delta w^{i-1} \quad \text{und} \quad w^i = w^{i-1} + \Delta w^i$$

mit einer kleinen positiven Lernrate η und einem *Impulsparameter* $\alpha \in [0, 1)$. Die so entstehende Folge $w^o, w^1, \ldots$ soll zu einem lokalen Minimum der Fehlerfunktion E konvergieren. □

Bemerkungen (2.1.8) (i) Die Methode des Gradientenabstiegs ist im Impulsverfahren enthalten: $\alpha = 0$.

(ii) Wird obige Rekursionsvorschrift für w^i expandiert, so ergibt sich der Ausdruck

$$w^i = w^{i-1} - \eta \nabla E|_{w^{i-1}} - \alpha\eta \nabla E|_{w^{i-2}} - \alpha^2 \eta \nabla E|_{w^{i-3}} - \ldots$$

Es ist unmittelbar einzusehen, daß $\alpha < 1$ sein muß, um ein geometrisches Wachsen der Folge $w^o, w^1, \ldots$ zu verhindern.

(iii) In einem Plateau der Fehlerfunktion mit nahezu konstantem Gradienten von kleinem Betrag beschleunigt ein großer Impulsparameter die Suche. Unter der Voraussetzung der Konstanz der Gradienten läßt sich obige Reihe aufsummieren,

$$w^i \approx w^{i-1} - \frac{\eta}{1-\alpha} \nabla E|_{w^{i-1}},$$

was eine effektive Lernrate von $\eta/(1-\alpha)$ bewirkt.

(iv) Die Hoffnung ist, daß im allgemeinen von der Impulsmethode der globale Trend der Änderung der Gewichtungsvektoren herauspräpariert wird.

Dies zeigt sich auch im konkreten Beispiel (siehe Abb. 23a). Leider ist hier ein weiterer Parameter einzustellen, der bei falscher Wahl die Konvergenz verschlechtern (siehe Abb. 23b) oder sogar zerstören kann. Bei eher zu klein gewählten Lernraten beschleunigt der Impulsanteil oft die Konvergenz. Sowohl von der theoretischen als auch von der praktischen Seite aus betrachtet erscheint der Einsatz des Impulsverfahrens eher unbefriedigend.

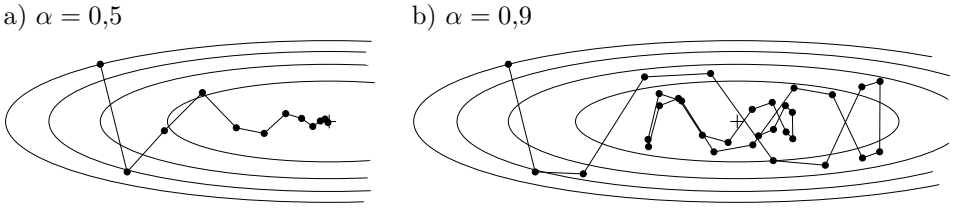


Abb. 23 Gradientenabstieg mit Impuls wie in Abb. 22 mit zwei verschiedenen Impulsparametern. a) Eine richtige Wahl des Impulsparameters zeigt den gewünschten Effekt der Glättung der Nähmaschinenschritte. b) Zu hohe Werte des Impulsparameters können schaden.

(v) Ursprünglich wurde die Impulsmethode im Online-Lernmodus (siehe Abschnitt 2.6) eingesetzt, wo sie auch die besseren Ergebnisse erzielt. □

2.2 Parameteradaption

In den vorherigen Analysen zeigte sich die Einstellung des Parameters η als kritisch für das Gelingen des Gradientenabstiegs; dies gilt umso mehr bei allgemeinen Fehlerfunktionen, die nicht die vereinfachte quadratische Form haben. Eine gute Idee scheint es zu sein, η im Verlauf des Lernvorgangs an die lokalen Gegebenheiten der Oberfläche der Fehlerfunktion anzupassen.

Parameteradaption nach Salomon (2.2.1) In ⟦Salomon 1991⟧ wird ein einfaches Verfahren zur *Parameteradaption* vorgestellt: Bei jeder Iteration des Backpropagation-Algorithmus werden zwei Testschritte, einmal mit verkleinerter (η_t/ζ) und einmal mit größerer Lernrate ($\eta_t\zeta$), durchgeführt und diejenige Lernrate mit kleinerem Fehler bevorzugt:

Wähle die anfängliche Lernrate $\eta_o > 0$. Setze mit einem Parameter $\zeta \geq 1$

$$\eta_{t+1} = \begin{cases} \eta_t \zeta & \text{falls } E(w - \eta_t \zeta \nabla E|_w) < E(w - (\eta_t/\zeta)\nabla E|_w) \\ \eta_t/\zeta & \text{sonst.} \end{cases}$$

Im Gegensatz zu η beeinflußt bei $\zeta > 1$ die konkrete Wahl von ζ nicht sonderlich das Konvergenzverhalten in unmittelbarer Nähe eines Minimums (allenfalls die konkrete Anzahl der benötigten Schritte). Typische Werte für ζ liegen im Bereich 1,2 bis 2,1. □

Bemerkungen (2.2.2) (i) Mit $\zeta = 1$ ist wieder die ursprüngliche Methode des Gradientenabstiegs mit konstanter Lernrate enthalten.

(ii) Für $\zeta > 1$ erweist sich das Verfahren als sehr wertvoll. Der im Tradeoff-Theorem (2.1.2) beschriebene Nachteil, daß der reine Backpropagation-Algorithmus bestimmte Minima der Fehlerfunktion nicht finden kann, wird durch die automatische Parameteradaption verhindert. In der Tat wurde dargelegt ⟦Salomon 1991⟧, daß für alle $\zeta > 1$ der Backpropagation-Algorithmus mit adaptiver Wahl von η bei quadratischer Fehlerfunktion konvergiert. Geschwindigkeitsvergleiche zeigen ⟦Moreira und Fiesler 1995, Salomon und van Hemmen 1996⟧, daß dieses Verfahren anderen Modifikationen von Backpropagation ebenbürtig oder überlegen ist.

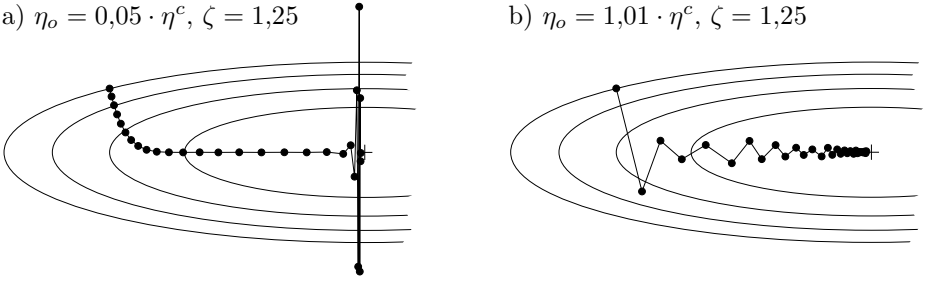


Abb. 24 Parameteradaption für die Lernrate mit dem Problem aus Abb. 21

(iii) Die Anpassung der Lernrate in einem Gebiet mit sich stark änderndem Gradienten kann ein schnelleres Reagieren erfordern als es die Parameteradaption (2.2.1) erlaubt. In Abb. 24a) beispielsweise wäre es kurz vor dem Ziel nötig gewesen, die Lernrate drastisch herunterzusetzen; in der Nähe

der w_1-Achse ändert sich sowohl die Richtung als auch der Betrag des Gradienten sehr stark. Dies kann jedoch erst nach mehreren Schritten geschehen, was einen Ausflug in Gebiete des Gewichtungsraums mit sehr großen Fehlerwerten zur Folge hat. Vielleicht waren es solche Effekte, die Salomon dazu bewogen haben, eine Adaption der Parameter mit Normierung des Gradienten vorzuschlagen: statt des Gradienten wird der normierte Gradient in (2.2.1) benutzt. Dieses Verfahren zeigt sich stabiler; jedoch gibt es auch hier die beschriebenen Effekte (siehe Abb. 25).

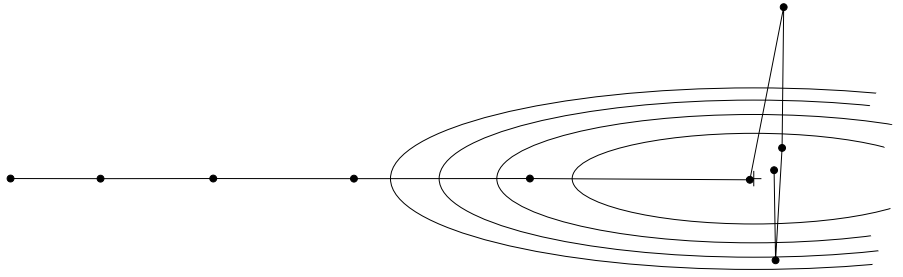


Abb. 25 Parameteradaption mit Normierung des Gradienten

(iv) In ⟦Salomon 1991⟧ werden Argumente dafür gegeben, daß bei Backpropagation mit Parameteradaption (2.2.1) mit oder ohne Normierung im quadratischen Fehlerfall nach einiger Zeit der Fehler unter den ursprünglichen Fehler fällt und danach für immer darunter bleibt. Die Möglichkeit, während des Lernvorgangs zwischenzeitlich schon mal beliebig große Fehler zu produzieren (wie in Abb. 24a und 25 demonstriert), führt unter anderem zu außerordentlich langen technischen Passagen in der Argumentation. Und es muß eine *im ganzen Gewichtungsraum* – statt z. B. auf $\{w \mid E(w) < E(w^o)\}$ – quadratische Fehlerfunktion vorausgesetzt werden!

(v) Die zeitweiligen Ausflüge des Gewichtungssatzes in weit vom Minimum entfernte Regionen schwächen den Wert der Beobachtung, daß für eine quadratische Fehlerfunktion das Minimum gefunden wird; die quadratische Näherung einer beliebigen in eine Taylorreihe entwickelbaren Fehlerfunktion ist nämlich nur in der Nähe eines regulären Minimums gültig. Nützlich wäre daher zumindest *asymptotische Stabilität*, d. h. daß gegen Ende des Lernvorgangs der Fehler nicht mehr ansteigt. Dies ist Gegenstand des nächsten Abschnitts. □

2.3 Asymptotisch stabile Parameteradaption

Eine Vergrößerung des Fehlers im Lauf des Lernens ist ein sicheres Anzeichen für eine zu große Lernrate in diesem Schritt. Dieses Anzeichen wird für geeignete „Gegenmaßnahmen“ benutzt.

Euler-Methode (2.3.1) Wird die diskrete Methode des Gradientenabstiegs ersetzt durch eine zeitkontinuierliche Version

$$\frac{d}{dt}w = -\eta \nabla E|_w,$$

so entstehen gekoppelte Differentialgleichungen für die nun von der Zeit t abhängigen Komponenten der Funktion $t \mapsto w(t)$. Die Abbildung

$$t \mapsto (w^o{}_1 \exp(-\eta t), w^o{}_2 \exp(-\lambda \eta t))$$

löst z. B. diese Differentialgleichungen mit dem Startwert $(w^o{}_1, w^o{}_2)$ für die Fehlerfunktion aus (2.1.5), siehe Abb. 26. Dabei schneidet die Kurve $t \mapsto w(t)$ die Höhenlinien jeweils senkrecht. η erweist sich in diesem Fall als ein unwichtiger Parameter, der nur die Zeit skaliert.

Die Methode des Gradientenabstiegs ist genau die *Euler-Methode*, obige Differentialgleichung zu lösen. Ein erheblicher Nachteil der Euler-Methode ist ihre Instabilität: jeder endliche Schritt verläßt die Trajektorie einer Lösung ohne die Bestrebung, wieder auf die Trajektorie zurückzukommen. Nahezu jeder andere Differentialgleichungs-Löser legt ein besseres Verhalten an den Tag; darunter gibt es auch einige, die mit dynamischer Parameteradaption arbeiten ⟦Press et al. 1988, Seiten 353–397⟧.

Jedoch erscheint im Kontext der Funktionsminimierung dieser Begriff der Stabilität zu stark. Es lohnt sich vor allem zu Beginn des Lernens nicht, viel Energie in das Bestreben zu stecken, genau auf einer bestimmten Lösungstrajektorie zu bleiben. Dies motiviert folgende Definition. □

Asymptotische Stabilität (2.3.2) Ein Minimierungsverfahren heiße *asymptotisch stabil*, wenn es am Ende des Minimierens – d. h. in einem Bereich um ein reguläres Minimum, in dem eine quadratische Approximation angenommen werden kann – nur noch Schritte unternimmt, die den Funktionswert verkleinern. □

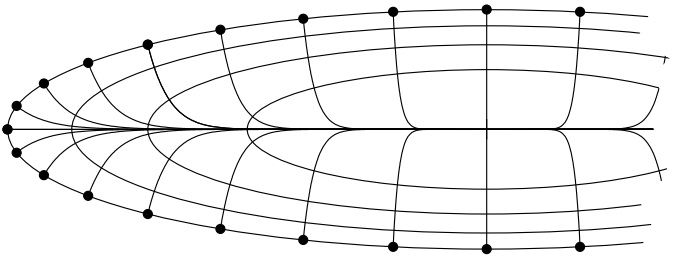


Abb. 26 Die einfache Fehlerfunktion $w \mapsto w_1^2/2 + \lambda w_2^2/2$ mit einer Schar von Lösungskurven für die Differentialgleichungen, die dem (kontinuierlichen) Gradientenabstieg entsprechen. Eingezeichnet sind jeweils Kurven $t \mapsto w(t)$ zu einem anfänglichen Startwert w^o (dicker Punkt) der Gewichte.

Asymptotisch stabile Parameteradaption (2.3.3) Sei $\zeta > 1$ beliebig, aber fest, und $\eta_o > 0$ (anfängliche Lernrate) beliebig. Wähle im t-ten Schritt des Backpropagation-Algorithmus einen beliebigen Vektor $g^t \in \mathbb{R}^E$, der nur ungefähr die gleiche Richtung wie $\nabla E(w^t)$ zu haben braucht und entweder einen beschränkten Betrag hat oder etwa wie $|\nabla E(w^t)|$ skaliert. Präziser formuliert: Seien $0 < a \leq 1 \leq b$ und $0 < c \leq 1$ beliebig, aber fest; wähle ein g^t mit $g^t \nabla E(w^t)/|g^t||\nabla E(w^t)| \geq c$ und gelte entweder für alle t, daß $1/|g^t| \in [a, b]$ oder für alle t, daß $|\nabla E(w^t)|/|g^t| \in [a, b]$. Sei noch die Abkürzung $e\colon \eta \mapsto E(w^t - \eta g^t)$ als eine durch E, g^t und w^t gegebene Fehlerfunktion von $\mathbb{R} \to \mathbb{R}$ definiert. Wiederhole (bis der Gradient verschwindet, ein oberes Limit für t oder ein unteres Limit $E_{\min}$ für den Fehler erreicht ist) die Iterationsvorschrift $w^{t+1} = w^t - \eta_{t+1} g^t$ mit

$$\eta_{t+1} = \begin{cases} \eta^* := \dfrac{\eta_t \zeta/2}{1 + \dfrac{e(\eta_t\zeta) - e(0)}{\eta_t \zeta g^t \nabla E(w^t)}} & \text{für } e(0) < e(\eta_t\zeta) \\ \eta_t/\zeta & \text{falls } e(\eta_t/\zeta) \leq e(\eta_t\zeta) \leq e(0) \\ \eta_t\zeta & \text{sonst.} \end{cases}$$

Dabei ist der erste Fall von η_{t+1} ein stabilisierender Term η^*, der bei quadratischer Fehlerfunktion (d. h. in Nähe eines Minimums) definitiv den Fehler verringert. η^* wird dann eingesetzt, wenn der Fehler $e(\eta_t\zeta)$, der im nächsten Schritt bei Wahl von $\eta_{t+1} = \eta_t\zeta$ entstünde, den Fehler $e(0)$ des augenblicklichen Gewichtungssatzes w^t übersteigt. □

Bemerkungen (2.3.4) (i) Vom eigentlichen Backpropagation-Algorithmus scheint kaum etwas übriggeblieben zu sein; jedoch ist auch hier immer noch ein zentraler Punkt, daß zu jedem Gewichtungssatz der Gradient des Fehlers bestimmt wird. Hieraus wird (wie auch immer) eine „ähnliche“ Richtung erzeugt und statt des Gradienten verwendet. Zum Beispiel erfüllt die Festlegung $g^t := \nabla E(w^t)$ oder etwa $g^t := \nabla E(w^t)/|\nabla E(w^t)|$ die Ähnlichkeitsforderung; der einzige Unterschied zu (2.2.1) ist dann die Verkleinerung der Schrittweite, falls $e(\eta_t\zeta) > e(0)$. Später (2.4.10) wird eine weitere Möglichkeit für Richtungen angegeben, die Zweite-Ordnung-Information benutzt.

(ii) Die Vektoren g^t sind kaum eingeschränkt: Die Richtungen müssen nur innerhalb eines Vorwärtskegel um den Gradienten mit beliebigem Öffnungswinkel kleiner als π liegen, und die Länge muß entweder absolut oder relativ zur Länge des Gradienten beschränkt sein (siehe Abb. 27).

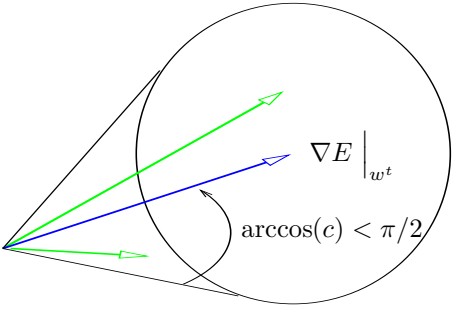


Abb. 27 Mögliche Richtungen des Fehlerabstiegs

(iii) Würde sich der Fehler beim Testschritt mit der größeren Schrittweite vergrößern, so wird durch die Konstruktion von η_{t+1} in (2.3.3) die neue Schrittweite η_{t+1} sicher kleiner als $\eta_t\zeta/2$ werden. Gegeben $\zeta < 2$, wird also sicher die Schrittweite unabhängig von der konkreten Fehlerfunktion verkleinert. Bei einer allgemeinen Fehlerfunktion kann es aber weit entfernt von einem Minimum vorkommen, daß der Fehler im nächsten Schritt ansteigt. Soll das nicht sein (z.B. weil die schärfere Forderung nach Stabilität statt nur asymptotischer Stabilität gestellt wird), dann könnte g^t in Richtung des Gradienten gewählt werden und durch sukzessive Verkleinerung der Schrittweite eine Verkleinerung des Fehlers bewirkt werden.

(iv) Bei der asymptotisch stabilen Parameteradaption (2.3.3) wird sich in (2.3.6) zeigen, daß der Fehler im quadratischen Fall im nächsten Schritt automatisch schon kleiner wird (daher die Bezeichnung asymptotisch stabil). Aber selbst dann muß sichergestellt werden, daß das Verfahren nicht „verhungert", d. h. daß die Verbesserungen nicht vorzeitig beliebig klein werden. Gelingt es beispielsweise zu zeigen, daß für ein festes $q < 1$ gilt $E(w^t) \leq qE(w^{t-1})$, so ist klar, daß der Fehler im Lauf der Zeit exponentiell zu Null zerfällt. Dies ist Inhalt des nächsten Abschnitts. □

Theorem der asymptotischen Konvergenz ⟦Rüger 1996b⟧ (2.3.5) *Sei eine positiv definite n-dimensionale quadratische Fehlerfunktion $w \mapsto E(w) = \sum \lambda_i {w_i}^2/2$ mit $\lambda_i > 0$ gegeben. Für alle $\zeta > 1$, $b \geq 1 \geq a > 0$ und $0 < c \leq 1$, beliebige anfängliche Schrittweiten $\eta_o > 0$ und Gewichte $w^o \in \mathbb{R}^n$ erzeugt die asymptotisch stabile Parameteradaption (2.3.3) eine monoton fallende Folge $t \mapsto w^t$, die mindestens exponentiell gegen das Minimum 0 der Fehlerfunktion konvergiert.* □

Beweis (2.3.6) O.B.d.A. sei $w^t \neq 0$ (angenommen, ein w_t wäre null: das ist genau dann der Fall, wenn $\nabla E(w^t) = 0$ und der Backpropagation-Algorithmus mit asymptotisch stabiler Parameteradaption terminiert ohnehin). Ziel ist es, zu zeigen, daß es ein $q < 1$ gibt mit

$$E(w^{t+1}) \leq qE(w^t)$$

für alle $w^t \neq 0$ und η_t, die sich durch alle möglichen $w^o \in \mathbb{R}^n$ und $\eta_o > 0$ ergeben können. Dann zerfällt die Folge $t \mapsto E(w^t)$ nämlich mindestens exponentiell zu Null. Da $w \mapsto \sqrt{E(w)}$ eine Norm im $\mathbb{R}^n$ ist, konvergiert mit gleicher Konvergenzgeschwindigkeit $t \mapsto w^t$ gegen das Minimum Null der Fehlerfunktion.

Fixiere w^t, η_t und ein gemäß den Voraussetzungen gewähltes g^t. Dann ist $e\colon \eta \mapsto E(w^t - \eta g^t)$ eine eindimensionale, konvexe quadratische Abbildung, da E eine positiv definite quadratische Form ist. Beachte, daß $e(0) = E(w^t)$ und $e(\eta_{t+1}) = E(w^{t+1})$. Habe $e(\eta)$ die Form $\alpha(\eta - \eta^*)^2 + \beta$, so lassen sich α, η^* und β durch $e(0)$, $\eta_t\zeta$, $e(\eta_t\zeta)$ und $e'(0) = -g^t\nabla E(w^t)$ ausdrücken: man überzeuge sich davon, daß das Minimum η^* von e durch den stabilisierenden Term η^* von (2.3.3) gegeben ist.

Ist nun $e(\eta_t\zeta) > e(0)$, so wird laut (2.3.3) η_{t+1} zu η^* gesetzt und w^{t+1} hat den kleinstmöglichen Fehler $E(w^{t+1}) = e(\eta^*) < e(0) = E(w^t)$ auf der Gerade, die durch g^t vorgegeben ist. Im anderen Fall $e(\eta_t\zeta) \leq e(0)$ können die drei verschiedenen Werte 0, $\eta_t\zeta$ und η_t/ζ nicht alle den gleichen Fehler haben, da e quadratisch ist. Derjenige Parameter wird als η_{t+1} bevorzugt, der den kleineren Fehler bewirkt und somit ist auch hier $E(w^{t+1}) < E(w^t)$. Für die gesuchte gleichmäßige Schranke q ist also schon nachgewiesen, daß $q \leq 1$. Der Rest des Beweises zeigt die Existenz einer Schranke $q < 1$.

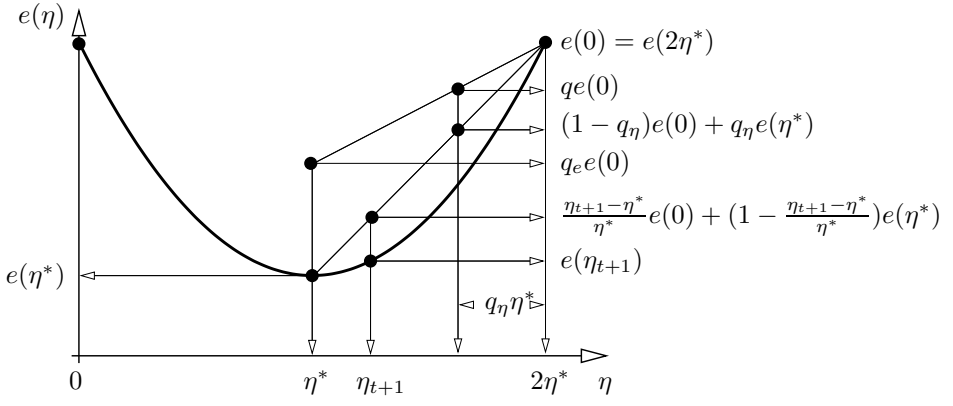


Abb. 28 Abschätzung einer Schranke q für die Verkleinerung des Fehlers

Wenn es zwei Konstanten $q_e, q_\eta \in (0,1)$ gibt mit

$$\eta_{t+1}/\eta^* \in [q_\eta, 2 - q_\eta] \quad \text{und} \tag{q_η}$$

$$e(\eta^*) \leq q_e e(0), \tag{q_e}$$

dann kann für $2\eta^* \geq \eta_{t+1} \geq \eta^*$ abgeschätzt werden, daß

$$\begin{aligned} e(\eta_{t+1}) &= e\Big(\frac{\eta_{t+1} - \eta^*}{\eta^*} 2\eta^* + \big(1 - \frac{\eta_{t+1} - \eta^*}{\eta^*}\big)\eta^*\Big) \\ &\leq \frac{\eta_{t+1} - \eta^*}{\eta^*} e(0) + \big(1 - \frac{\eta_{t+1} - \eta^*}{\eta^*}\big) e(\eta^*) \\ &\leq (1 - q_\eta) e(0) + q_\eta e(\eta^*) \\ &\leq (1 - q_\eta(1 - q_e)) e(0). \end{aligned}$$

Die erste Ungleichung folgt aus der Konvexität von e, die zweite benutzt die Eigenschaft (q_η) und die dritte (q_e). Abb. 28 veranschaulicht die Abschätzungen. Für den spiegelbildlichen Fall $0 < \eta_{t+1} \leq \eta^*$ ergibt sich die gleiche Abschätzung

$$E(w^{t+1}) \leq \underbrace{(1 - q_\eta(1 - q_e))}_{q < 1} E(w^t).$$

Werte für η_{t+1} in $\mathbb{R}\setminus(0, 2\eta^*)$ sind nicht möglich, da sonst $E(w^{t+1}) \geq E(w^t)$, was schon ausgeschlossen war. Es sind also nur noch Abschätzungen für q_e und q_η zu finden.

Zu (q_e): Durch quadratische Ergänzung wird e als $e(\eta) = E(w - \eta g) = \alpha(\eta - \eta^*)^2 + e(\eta^*)$ geschrieben mit

$$\eta^* = \frac{\sum_i \lambda_i w_i g_i}{\sum_i \lambda_i {g_i}^2} \quad \text{und} \quad e(\eta^*) = e(0) - \frac{\eta^*}{2} \sum_i \lambda_i w_i g_i. \qquad (\eta^*)$$

Es folgt

$$\begin{aligned} \frac{e(\eta^*)}{e(0)} &= 1 - \frac{\big(g \nabla E(w)\big)^2}{\sum_i \lambda_i {g_i}^2 \sum_i \lambda_i {w_i}^2} \\ &\leq 1 - c^2 \frac{\sum_i {g_i}^2}{\sum_i \lambda_i {g_i}^2} \frac{\sum_i {\lambda_i}^2 {w_i}^2}{\sum_i \lambda_i {w_i}^2} \\ &\leq \underbrace{1 - c^2 \frac{\lambda^<}{\lambda^>}}_{q_e < 1} \end{aligned}$$

mit $\lambda^< := \min_i\{\lambda_i\}$ und $\lambda^> := \max_i\{\lambda_i\}$. Die erste Ungleichung ergibt sich aus der Voraussetzung, daß $g \nabla E(w)/|g||\nabla E(w)| \geq c$. Die zweite folgt aus der Beobachtung, daß die beteiligten Brüche Konvexkombinationen von λ_i bzw. $1/\lambda_i$ sind, die durch $\lambda^<$ bzw. $1/\lambda^>$ minimiert werden.

Zu (q_η): Die Bedingung $e(\eta/\zeta) \leq e(\eta\zeta)$ ist äquivalent mit $(\eta/\zeta - \eta^*)^2 \leq (\eta\zeta - \eta^*)^2$ und dies wiederum mit $\eta \geq 2\eta^*\zeta/(\zeta^2 + 1)$. Daher lautet im quadratischen Fehlerfall die asymptotisch stabile Parameteradaption

$$\eta_{t+1} = \begin{cases} \eta^* & \text{für } \eta_t > 2\eta^*/\zeta, \\ \eta_t/\zeta & \text{falls } 2\eta^*\zeta/(\zeta^2 + 1) \leq \eta_t \leq 2\eta^*/\zeta \text{ und} \\ \eta_t\zeta & \text{ansonsten.} \end{cases}$$

Die optimale Schrittweite hängt sowohl vom Ort w^t als auch der gerade gewählten Richtung g^t ab; dies wird zur Verdeutlichung als $\eta^*(w^t, g^t)$ notiert. Wegen obiger Regel für die Parameteradaption kann die induktive Zugehörigkeit

$$\frac{\eta_{t+1}}{\eta^*(w^t,g^t)} \in \begin{cases} \left[\min\left\{\frac{2}{\zeta^2+1}, \frac{\eta_t}{\eta^*(w^{t-1},g^{t-1})} \cdot \frac{\eta^*(w^{t-1},g^{t-1})}{\eta^*(w^t,g^t)}\zeta\right\}, z\right] & \text{für } t > 0 \\ \left[\min\left\{\frac{2}{\zeta^2+1}, \frac{\eta_o}{\eta^*(w^o,g^o)}\zeta\right\}, z\right] & \text{für } t = 0 \end{cases}$$

mit $z := \max\{2/\zeta^2, 2\zeta^2/(\zeta^2+1)\}$ angegeben werden. Durch sorgfältiges Zurückführen auf den Induktionsanfang ergibt sich für alle $t \geq 0$

$$\frac{\eta_{t+1}}{\eta^*(w^t,g^t)} \in \left[\min_{0\leq s\leq t}\left\{\frac{2}{\zeta^2+1}\zeta^s \frac{\eta^*(w^{t-s},g^{t-s})}{\eta^*(w^t,g^t)}, \zeta^{t+1}\frac{\eta_o}{\eta^*(w^t,g^t)}\right\}, z\right].$$

Nun ist wegen (η^*) aber auch

$$\begin{aligned} \eta^*(w,g) &= \frac{g\nabla E(w)}{|g||\nabla E(w)|} \frac{|g||\nabla E(w)|}{\sum_i \lambda_i {g_i}^2} \\ &= \frac{g\nabla E(w)}{|g||\nabla E(w)|} \frac{1}{\sum_i \lambda_i \frac{{g_i}^2}{\sum_j {g_j}^2}} \frac{|\nabla E(w)|}{|g|}. \end{aligned}$$

Wenn g immer so gewählt ist, daß $|\nabla E(w)|/|g| \in [a,b]$, ist η^* gleichmäßig beschränkt mit $\eta^* \in [ca/\lambda^>, b/\lambda^<]$. Daher kann in diesem Fall die gesuchte Schranke q_η gesetzt werden zu

$$q_\eta := \min\left\{\frac{2}{\zeta^2+1}, \frac{2\zeta}{\zeta^2+1}\frac{ca}{b}\frac{\lambda^<}{\lambda^>}, \frac{\eta_o\lambda^<\zeta}{b}\right\} > 0.$$

Für den Fall, daß $1/|g|$ beschränkt ist, kann ausgenutzt werden, daß

$$|\nabla E(w)|^2 = 2\sum_i \lambda_i \frac{\frac{1}{2}\lambda_i {w_i}^2}{\sum_j \frac{1}{2}\lambda_j {w_j}^2} E(w).$$

Daher liegt $\eta^*(w,g)$ im Bereich $[ca\sqrt{2\lambda^< E(w)}/\lambda^>, b\sqrt{2\lambda^> E(w)}/\lambda^<]$. Da $t \mapsto E(w^t)$ monoton fällt, kann q_η abgeschätzt werden zu

$$q_\eta := \min\left\{\frac{2}{\zeta^2+1}, \frac{2\zeta}{\zeta^2+1}\frac{ca}{b}\left(\frac{\lambda^<}{\lambda^>}\right)^{3/2}, \frac{\eta_o\lambda^<\zeta}{b\sqrt{2\lambda^> E(w^o)}}\right\} > 0. \qquad \blacksquare$$

2.4 Verfahren zweiter Ordnung

Newton-Verfahren (2.4.1) Sei w^* ein lokales Minimum einer in eine Taylorreihe entwickelbaren Fehlerfunktion E. Von der um w^o entwickelten Taylorreihe von E kann dann der Gradient bezüglich w gebildet werden:

$$\nabla E(w) = \nabla E(w^o) + H|_{w^o}\,(w - w^o) + \ldots$$

Insbesondere gilt diese Expansion für das Minimum $w = w^*$, wo der Gradient verschwindet. Unter Weglassen der höheren Terme ergibt sich die ungefähre Beziehung $w^* \approx w^o - H^{-1}|_{w^o}\,\nabla E|_{w^o}$, die nur für quadratisches E exakt gilt. Für andere E ist die Hoffnung, daß die daraus gewonnene Iterationsvorschrift

$$w^{i+1} = w^i - H^{-1}|_{w^i}\,\nabla E|_{w^i}$$

zum gewünschten Minimum konvergiert. Hier wird deutlich, daß statt der reellen Lernrate η in der Methode des Gradientenabstiegs die Linksmultiplikation mit der inversen Hesseschen Matrix optimal wäre, *wenn* w^i schon in der Nähe eines Minimums ist. Dann nämlich wird quadratische Konvergenz erreicht (die Genauigkeit, d. h. die Anzahl der korrekten Stellen des Minimums wächst quadratisch mit der Zahl der Iterationen) statt linearer Konvergenz (z.B. bei exponentiell zerfallendem Fehler).

Die Komplexität der Berechnung der Inversen einer Matrix ist die gleiche wie für eine Matrixmultiplikation und es existieren effiziente Algorithmen zur Berechnung der Hesseschen Matrix einer durch ein neuronales Netz gegebenen Fehlerfunktion ⟦Hassibi und Stork 1993, Buntine und Weigend 1994⟧. Mit den Techniken von Kapitel 1 kann aber auch schnell ein eigener Algorithmus abgeleitet werden: z. B. kann das Zurückpropagieren mit der Berechnung der Deltas selbst als Abbildungsnetz aufgefaßt werden, auf das wiederum die Kettenregel (1.1.8) angewendet wird.

Das einzige praktische Problem scheint zunächst bei großen Netzen der Speicherplatzbedarf für die Hessesche Matrix zu sein. Andererseits ist in obige Herleitung gar nicht die volle Minimumeigenschaft eingegangen: die gleichen Argumente hätten bei der Maximumsuche zur gleichen Formel geführt. Zudem ist zwar in einer ganzen Umgebung eines typischen Minimums die Hessesche Matrix positiv definit, was etwas entfernt schon ganz

anders aussehen kann, und schon gar nicht für beliebige Startwerte w^o gelten muß. Selbst wenn im gesamten Definitionsbereich die Hessesche Matrix positiv definit – und damit umkehrbar – ist, wie z. B. bei der eindimensionalen Funktion $w \mapsto \int_1^w \log(x)dx$ auf $\mathbb{R}_o^+$, kann das Newton-Verfahren instabil sein; in diesem Beispiel ist $w^{i+1} = w^i(1 - \log(w^i))$, was für Startwerte $w^o > \exp(1)$ Probleme bereitet (siehe Abb. 29).

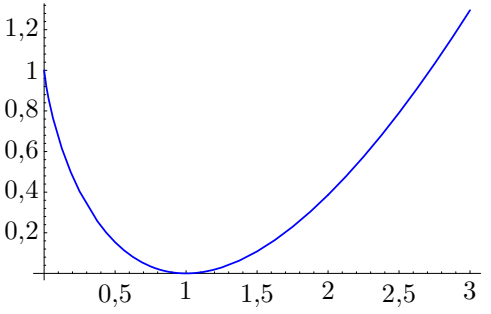


Abb. 29 Die Abbildung $w \mapsto \int_1^w \log(x)dx$; das Newton-Verfahren verläßt für Startwerte größer als $\exp(1)$ sofort den Definitionsbereich $\mathbb{R}_o^+$

Liniensuche (2.4.2) Obwohl selbst kein Verfahren zweiter Ordnung, wird es doch oft als Bestandteil solcher Verfahren eingesetzt. Die eindimensionale Abbildung $\eta \mapsto E(w^i - \eta g^i)$ wird minimiert, wobei g^i eine bestimmte Richtung ist. Die Stelle des Minimums η_i wird in

$$w^{i+1} = w^i - \eta_i g^i$$

zur Iteration der Gewichte benutzt. Eine ganz naheliegende Wahl für die Richtung g^i ist der Gradient $\nabla E|_{w^i}$; dieses Verfahren heißt dann *Gradientenliniensuche* (engl. *steepest descent line search*). □

Bemerkungen (2.4.3) (i) Die Parameteradaption (2.2.1) ist sehr verwandt mit diesem Vorgehen, wobei nicht lange ein Minimum der eindimensionalen Funktion präzise gesucht wird, sondern nur grob eine gute Schrittweite geschätzt wird; dabei anfallende Fortschritte werden sofort umgesetzt.

(ii) Bei der Gradientenliniensuche stehen notwendigerweise aufeinanderfolgende Suchrichtungen senkrecht zueinander, denn im Minimum der Liniensuche in Richtung g^i verschwindet die partielle Ableitung

$$\partial(\eta \mapsto E(w^i - \eta g^i))/\partial\eta|_{\eta_i} = -\nabla E|_{w^{i+1}}^T g_i.$$

Dieses Verhalten ist auch in Abb. 30 ersichtlich. □

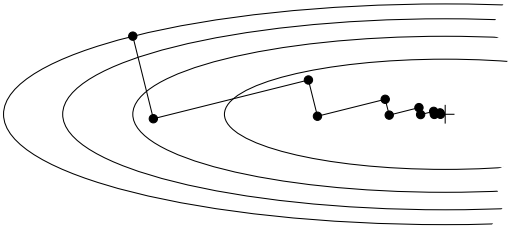


Abb. 30 Gradientenliniensuche

Methode der konjugierten Gradienten (2.4.4) Die Gradientenliniensuche scheint etwas unbeholfen, selbst in zwei Dimensionen das Minimum einer quadratischen Fehlerfunktion $E\colon \mathbb{R}^n \to \mathbb{R}$ zu finden. In gewisser Weise sollte die neue Suchrichtung für die Liniensuche besser der Gleichung

$$g^{iT} \nabla E|_{w^{i+1} - \eta g^{i+1}} = 0$$

genügen; denn dann stört eine Optimierung in Richtung g^{i+1} möglichst wenig einen bereits erzielten Fortschritt zum Auf-Null-bringen des Gradienten bezüglich der Richtung g^i. Mit der Taylor-Approximation für den Gradienten ∇E um w^i aus (2.4.1) zusammen mit $g^{iT} \nabla E_{w^{i+1}} = 0$ aus (2.4.3;ii) gilt dann in erster Ordnung

$$g^{iT} H|_{w^{i+1}} g^{i+1} = 0.$$

Zwei Vektoren g^i und g^{i+1}, die dieser Bedingung genügen, heißen *konjugiert* zueinander bezüglich H. Ziel bei der Methode der konjugierten Gradienten ist es jetzt, n Suchrichtungen anzugeben, die paarweise konjugiert zueinander sind. Die *Polak-Ribière-Richtungen*

$$g^i = \begin{cases} \nabla E|_{w^o} & \text{falls i=0} \\ \nabla E|_{w^i} + \beta g^{i-1} & \text{sonst mit } \beta = \dfrac{(\nabla E|_{w^i} - \nabla E|_{w^{i-1}})^T \nabla E|_{w^i}}{(\nabla E|_{w^{i-1}})^2} \end{cases}$$

erfüllen diese Forderungen für eine quadratische Fehlerfunktion E. Zusätzlich führt dann die Iteration vom Punkt w^i zum Punkt w^{i+1} durch Minimumsuche in Richtung g^i nach spätestens n Schritten zum Minimum. Das ist der Inhalt von Korollar (2.4.7) und wird in Abb. 31 für $n = 2$ demonstriert.

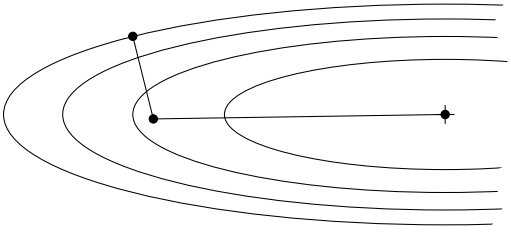


Abb. 31 Methode der konjugierten Gradienten

Für eine allgemeine strikt konvexe Fehlerfunktion können nun trotzdem nach obiger Methode n Schritte durchgeführt werden, um näher an das Minimum zu kommen. Dann wird das Verfahren resettet ($i = 0$ gesetzt mit $w^o \leftarrow w^n$). Solche Zyklen bestehend aus n Schritten werden solange wiederholt bis das Minimum mit akzeptabler Genauigkeit erreicht ist; die Konvergenzgeschwindigkeit ist hierbei wie beim Newton-Verfahren quadratisch ⟦Cohen 1970, Polak 1971 (Kapitel 6.3)⟧. □

Theorem für konjugierte Gradienten (2.4.5) *Sei H eine symmetrische, positiv definite $n \times n$-Matrix und $\nabla^o \in \mathbb{R}^n$ beliebig. Mit der Verankerung $g^o = \nabla^o$ sei für $i \geq 0$*

$$g^{i+1} := \nabla^{i+1} + \beta_i g^i \quad \textit{mit} \quad \beta_i := |\nabla^{i+1}|^2 / |\nabla^i|^2 \tag{g}$$

$$\nabla^{i+1} := \nabla^i - \eta_i H g^i \quad \textit{mit} \quad \eta_i := {g^i}^T \nabla^i / {g^i}^T H g^i, \tag{∇}$$

wobei ggf. die Faktoren η_i oder β_i je auf Null zu setzen sind, wenn der entsprechende Nenner verschwindet. Dann gelten Orthogonalitäts- und Konjugiertheitseigenschaften, nämlich daß für alle $0 \leq i < j$

$${\nabla^i}^T \nabla^j = {g^i}^T \nabla^j = {g^i}^T H g^j = 0. \qquad \square$$

Beweis (2.4.6) Durch sorgfältige Induktion (siehe z. B. ⟦Polak 1971, Rüger 1996a⟧). Im wesentlichen ist das obige Schema eine Variante des Schmidtschen Orthogonalisierungsverfahrens. ∎

Korollar (2.4.7) *Das Theorem kann auf die Methode der konjugierten Gradienten für eine positiv definite quadratische Form $E = w \mapsto w^T H w$ angewendet werden. Dabei wird ∇^i mit $\nabla E|_{w^i}$ identifiziert; w^n erweist sich als das Minimum von E.* □

Beweis (2.4.8) Die Rekurrenz (2.4.5;∇) ist gerade die Taylorentwicklung von $\nabla|_{w^i - \eta_i g^i}$ um den Punkt w_i unter Respektierung von $w^{i+1} = w^i - \eta_i g^i$. Die Extremal-Bedingung $\partial\eta \mapsto E(w^i - \eta g^i)/\partial\eta|_{\eta_i} = 0$ kann nach η_i aufgelöst werden und es ergibt sich gerade die Formel (2.4.5;∇). Soll β aus (2.4.4) mit β_{i-1} aus (2.4.5;g) identifiziert werden – und daher auch die Folgen $i \mapsto g^i$ miteinander –, so müßte ∇^{i+1} auf ∇^i senkrecht stehen.

Aber das ist ein Teil der Aussage des dann zutreffenden Theorems für konjugierte Gradienten, daß nämlich alle Gradienten ∇^i *paarweise* senkrecht aufeinander stehen. Da in $\mathbb{R}^n$ höchstens n Vektoren paarweise zueinander senkrecht sein können ohne null zu sein, muß spätestens ∇^n verschwinden, d. h. bei w^n ein Minimum vorliegen (ist ein ∇^k null, so auch alle ∇^j mit $j > k$, wie leicht der Definition zu entnehmen ist). ∎

Bemerkung (2.4.9) Wie sich im obigen Korollar gezeigt hat, verschwindet bei quadratischer Fehlerfunktion der Term ${\nabla^{i+1}}^T \nabla^i$, mithin hätte bei der Methode der konjugierten Gradienten β auch ohne diesen Term definiert sein können; das ist in der Tat die Version mit *Fletcher-Reeves-Richtungen*. Die später von Polak und Ribière eingefügte Modifikation stabilisiert die Iteration, wenn die Fehlerfunktion nicht quadratisch ist, oder wenn das Minimum bei der Liniensuche nicht genau bestimmt wurde ⟦Polak und Ribière 1969, Jacobs 1977 (III.1.7)⟧. □

Parameteradaption mit normierten Polak-Ribière-Richtungen (2.4.10) Es spricht nichts dagegen, die asymptotisch stabile Parameteradaption (2.3.3) mit normierten Polak-Ribière-Richtungen auszustatten: Ein Vektor g^i wird mit der Rekurrenz aus (2.4.4) berechnet, und $g_i/|g^i|$ als Vektor für den Fehlerabstieg in (2.3.3) benutzt.

Bei jedem Schritt sollte geprüft werden, ob die Projektion des normierten Polak-Ribière-Vektors $g^i/|g^i|$ auf den Gradienten $\nabla E|_{w^i}$ eine Länge von mindestens einer Konstante $c \in]0, 1]$ besitzt; falls nicht, sollte das Verfahren resettet werden. Dann erfüllt nämlich diese Technik die Voraussetzungen des Theorems der asymptotischen Konvergenz (2.3.5). In einer praktischen Implementierung aber stört es nicht, $c = 0$ zu verwenden.

Die Idee ist, daß über die Polak-Ribière-Richtungen Informationen zweiter Ordnung in das Lernverfahren einfließen, ohne daß die aufwendige Minimierung der Liniensuche exakt gemacht werden muß. □

2.5 Vergleich und Grenzen der Verfahren

Aus der Literatur sind schon Vergleiche einiger Lernverfahren bekannt, z. B. ⟦Salomon 1991, Pfister und Rojas 1993, Moreira und Fiesler 1995, Salomon und van Hemmen 1996, Pfister und Rojas 1996⟧. In diesen Referenzen ist die eine oder andere Form der Parameteradaption (2.2.1) jeweils mit anderen Verfahren verglichen worden.

Nun hat die asymptotisch stabile Parameteradaption (2.3.3) kein wesentlich anderes Laufzeitverhalten als die Parameteradaption (2.2.1), was der Unterabschnitt (2.5.2) belegt. Damit ist die Einordnung des neuen Verfahrens (2.3.3) auf andere zurückgeführt. Der Unterabschnitt (2.5.2) zeigt außerdem, wie die Einbeziehung von Zweiter-Ordnung-Information durch Benutzung von Polak-Ribière-Richtungen (2.4.10) das Lernen beschleunigt. Damit aber eine Einordnung der verschiedenen Verfahren innerhalb dieser Arbeit erfolgen kann, wurden auch der traditionelle Backpropagation-Algorithmus und die Methode der konjugierten Gradienten anhand einiger Probleme untersucht.

Die untersuchten Probleme (2.5.1) Die in diesem Abschnitt studierten Probleme sind i. w. Funktionsapproximationen.

(a) *Regression*: Mit einem 3-2-4-Netz (drei Eingabe-, zwei Binnen- und vier Ausgabeknoten) und einem bestimmten Gewichtungssatz wurden Pattern-Target-Beispiele generiert und die Targets verrauscht. Ein Netz gleicher

Struktur soll dann mit einem zufälligen anfänglichen Gewichtungsvektor die Daten approximieren lernen.

(b) *Approximation*: wie (a), nur ohne Verrauschen der Targets.

(c), (d) und (e): Diese Probleme sind durch die *künstliche Fehlerfunktion*

$$w \mapsto E(w) = \sum_{i=1}^{n=76} i|w_i|^p \qquad (E)$$

mit verschiedenen Potenzen gegeben. In (c) ist die Fehlerfunktion rein quadratisch. Andere Potenzen als Zwei, wie sie z. B. in den folgenden Tabellen bei Problemen (d) und (e) auftauchen, sollten auch approximierbar sein, solange sie nur größer als etwa 1,5 sind. Die Algorithmen in diesem Kapitel verlassen sich nämlich darauf, daß der Gradient im Minimum verschwindet, was für eine Potenz wie 1,1 numerisch fragwürdig ist. Potenzen kleiner als 1 erzeugen eine nicht mehr konvexe Fehlerfunktion; diese werden in Unterabschnitt (2.5.6) kommentiert. Nun ist aber das Verhalten eines typischen Minimums eher durch Potenzen wie $2, 4, \ldots$ gegeben; daher wird bei Problem (e) die Potenz 4 untersucht.

(f) *Enkoder*: Das Problem des 8-3-8-Enkoders besteht aus dem Lernen der Identität, eingeschränkt auf die acht Einheitsvektoren des $\mathbb{R}^8$; dabei bildet sich in der Binnenschicht typischerweise eine Binärkodierung der acht Eingabemuster aus. Dieses Problem wurde studiert, da die zugehörige Fehlerfunktion globale Minima am Rand des Definitionsbereichs hat: Die gewünschten Werte 0 und 1 an den Ausgabeknoten können von der Fermifunktion nur asymptotisch erreicht werden; demzufolge muß mindestens eines der Gewichte unendlichen Betrag haben. Solche Minima sind nicht vom Theorem der asymptotischen Konvergenz abgedeckt und werden daher hier empirisch untersucht. □

Epochenanzahl der Parameteradaptionsverfahren (2.5.2) In Tabelle 1 wird für die Parameteradaption nach Salomon mit normiertem Gradienten (2.2.2;iii), die asymptotisch stabile Parameteradaption (2.3.3) mit normiertem Gradienten und für die asymptotisch stabile Parameteradaption mit normierten Polak-Ribière-Richtungen (2.4.10) jeweils die mittlere

Anzahl der Epochen (inklusive empirischer Standardabweichung) angegeben, die nötig war, um eine bestimmte Fehlerschranke $E_{\min}$ zu unterschreiten. Die Mittelung erfolgte über $\zeta \in [1{,}7,\ 2{,}1]$ und anfängliche Gewichtungsvektoren mit in $[-1, 1]$ gleichverteilten Komponenten.

Die Tabelle 1 zeigt dabei, daß die Stabilisierung der ursprünglichen Parameteradaption selbst noch keinen erheblichen Geschwindigkeitsgewinn mit sich bringt. Bei den künstlichen Fehlerfunktionen ist das stabilisierte Verfahren sogar langsamer: Dies läßt sich dadurch erklären, daß die Stabilisierung die Schrittweite in einer Kurve der Lösungstrajektorie deutlich verkleinert. Die ursprüngliche Parameteradaption fliegt zwar aus der Kurve (wie in Abb. 24a und Abb. 25 demonstriert), wird aber durch die Regelmäßigkeit der künstlichen Fehlerfunktionen weit entfernt vom Minimum relativ schnell wieder zurückgeholt. Diese Struktur ist bei realistischeren Optimierungsproblemen, wie (a), (b) und (f), nicht gegeben, weswegen dort die stabilisierte Variante leicht im Vorteil ist.

Deutliche Geschwindigkeitssteigerungen bringt die Verwendung von Polak-Ribière-Richtungen mit sich. Dies ist auch schon deswegen zu erwarten, weil Zweite-Ordnung-Information einfließt. Nur im Fall der künstlichen Fehlerfunktion mit Potenz 4 (also bei einem Minimum mit verschwindender Hessescher Matrix) sind die Polak-Ribiere-Richtungen eher schädlich als nützlich. Aber durch die wirklich starke Ausprägung des Minimums sind alle Parameteradaptionsverfahren vergleichbar schnell am Ziel angelangt.

In Tabelle 1 wird für das Enkoder-Problem auch demonstriert, daß der Rand des Gewichtungsraums $\mathbb{R}^n$ in wenigen Schritten erreicht wird; dabei ist entscheidend, daß die Lernrate geometrisch anwachsen kann. □

Epochenanzahl zweier traditioneller Verfahren (2.5.3) Nach dem Trade-off-Theorem für Backpropagation hängt die Konvergenz des reinen Backpropagation-Algorithmus davon ab, daß die feste Lernrate richtig eingestellt ist. In einigen Versuchsläufen wurde für die untersuchten Probleme empirisch eine geeignete Lernrate η_{opt} bestimmt. Tabelle 2 zeigt in der Spalte BP die durchschnittliche Epochenanzahl von Backpropagation mit dieser speziellen Lernrate. Das Problem (e) hat ein Minimum, dessen Hessesche Matrix verschwindet; es läßt sich mit einer festen Lernrate nicht lösen.

Tabelle 1 Mittlere Epochenanzahl der Parameteradaptionsverfahren

Problem	$E_{\min}$	(2.2.2;iii)	(2.3.3)	(2.4.10)
(a) 3-2-4-Regression	10^{0}	$101 \pm 50\%$	$98 \pm 49\%$	$37 \pm 39\%$
(b) 3-2-4-Approximation	10^{-4}	$550 \pm 70\%$	$535 \pm 70\%$	$106 \pm 50\%$
(c) Quadratisch ($n = 76$)	10^{-16}	$440 \pm 14\%$	$485 \pm 20\%$	$115 \pm 9\%$
(d) Potenz 1,8 ($n = 76$)	10^{-4}	$380 \pm 25\%$	$456 \pm 25\%$	$80 \pm 22\%$
(e) Potenz 4 ($n = 76$)	10^{-16}	$24 \pm 10\%$	$27 \pm 11\%$	$33 \pm 13\%$
(f) 8-3-8-Enkoder	10^{-3}	$795 \pm 55\%$	$775 \pm 55\%$	$200 \pm 70\%$
Aufwand pro Epoche		1,5	1,5	1,5

Tabelle 2 Mittlere Epochenanzahl zweier traditioneller Verfahren

Problem	$E_{\min}$	η_{opt}	BP mit η_{opt}	PR
(a) 3-2-4-Regression	10^{0}	0,2	$354 \pm 50\%$	$26 \pm 34\%$
(b) 3-2-4-Approximation	10^{-4}	1,0	$900 \pm 70\%$	$62 \pm 41\%$
(c) Quadratisch ($n = 76$)	10^{-16}	2/77	$735 \pm 4\%$	76
(d) Potenz 1,8 ($n = 76$)	10^{-4}	0,005	$530 \pm 24\%$	–
(e) Potenz 4 ($n = 76$)	10^{-16}	0	∞	–
(f) 8-3-8-Enkoder	10^{-3}	2,0	$21.400 \pm 9\%$	$190 \pm 88\%$ ♮
Aufwand pro Epoche			1	3–5

♮ 17% der Versuche konvergierten nicht

Es zeigt sich hier, daß alle Parameteradaptionsverfahren weniger Epochen benötigen als der mühsam besteingestellte Backpropagation-Algorithmus. Dies ist besonders dort auffällig, wo für den Gewichtungsvektor weite Wege zurückzulegen sind: Beim Enkoder-Problem liegen nämlich die Optima der Fehlerfunktion am Rand von $\mathbb{R}^n$. Auf dem Weg zu einem Optimum wären im Lauf des Lernens unterschiedliche Lernraten angebracht. In ⟦Salomon und van Hemmen 1996⟧ werden solche Lernprobleme intensiv studiert und zum Vergleich mit anderen Verfahren herangezogen (4-2-4-, 8-3-8-, 10-5-10-, 16-4-16-, 32-5-32- und 64-6-64-Enkoder, 2-2-1- und 4-4-1-Parität und 3-1-8-8-Dekoder).

In Tabelle 2 ist in der Spalte PR auch für die praktisch relevanten Fälle (a), (b) und (f) jeweils die Anzahl der Epochen für die Methode der konjugierten Gradienten mit Polak-Ribière-Richtungen dargestellt. Der verwendete Simulator erlaubte keine künstlichen Fehlerfunktionen; daher kann nur für (c) das aus Korollar (2.4.7) bekannte theoretische Resultat angegeben werden. Die Methode der konjugierten Gradienten benötigt in der Regel weniger Epochen als die verwandte Methode der stabilen Parameteradaption mit Polak-Ribière-Richtungen, jedoch ist der Rechenaufwand pro Epoche deutlich größer wie im folgenden ausgeführt wird. □

Aufwand pro Epoche (2.5.4) Alle in diesem Kapitel untersuchten Algorithmen setzen den Backpropagation-Algorithmus ein, um den Wert der Fehlerfunktion (Propagieren) und zugleich den Wert des Gradienten (beim Zurückpropagieren) an einer Stelle im Gewichtungsraum zu bestimmen. Der Zeitaufwand für beide Berechnungsanteile ist annähernd gleich – siehe (1.3.5) – und soll hier jeweils mit 1/2 angesetzt werden. Damit hat der Backpropagation-Algorithmus den Aufwand 1 pro Epoche.

Die Parameteradaption macht in jedem Zyklus zwei Testschritte. Da der zweite, erfolgreiche Testschritt der Parameteradaption bei effizienter Implementierung schon als der Propagierungs-Schritt des nächsten Zyklus zur Berechnung des Gradienten wieder benutzt werden kann, hat die Parameteradaption etwa den Aufwand 1,5 pro Epoche. Der Zeitaufwand zu einer eventuellen Normierung ist unerheblich.

Im Prinzip gilt das auch für die asymptotisch stabile Parameteradaption. Wird aber gemäß der Fallunterscheidung erkannt, daß der Fehler bei Verwendung der größeren Lernrate $\eta_t\zeta$ ansteigen würde, dann braucht der zweite Testschritt nicht mehr gemacht werden. Der Zeitaufwand zur Berechnung der Polak-Ribière-Richtungen kann vernachlässigt werden.

Die Methode der konjugierten Gradienten hat einen zunächst unbekannten Aufwand pro Epoche für die nötige Liniensuche. Es ist jedoch plausibel, daß dafür mindestens etwa 4 bis 8 Fehlerfunktionsauswertungen gebraucht werden. Damit entsteht zusammen mit der Gradientenbestimmung ein Aufwand von 3 bis 5 pro Epoche. Der Aufwandsfaktor, mit dem die

Epochenanzahl für einen gerechten Laufzeitvergleich multipliziert werden müßte, ist in den Tabellen 1 und 2 jeweils in der letzten Zeile angegeben. □

Abhängigkeit von der Gewichtungsinitialisierung (2.5.5) Den absoluten Zahlen in den Tabellen 1 und 2 kommt eine untergeordnete Bedeutung zu. Sie hängen zum Beispiel auch noch von der Wahrscheinlichkeitsverteilung der anfänglichen Gewichtungsvektoren ab: In obigen Tabellen wird über Experimente gemittelt, bei denen die Komponenten der Gewichtungsvektoren zu Beginn gleichverteilt in $[-1, 1]$ waren. Die Standardabweichung der Komponenten ist hier $1/\sqrt{3} \approx 0{,}58$ und die Gewichtungsvektoren stammen aus einem kompakten Gebiet.

In ⟦Rüger 1996b⟧ wurden Ergebnisse sehr ähnlicher Experimente, aber diesmal mit standard-normalverteilten Komponenten, veröffentlicht (siehe Tabelle 3). Hier sind die anfänglichen Gewichtungsvektoren im Prinzip aus dem ganzen $\mathbb{R}^n$ und die Standardabweichung der Komponenten ist 1. Schon alleine wegen der größeren Variabilität der anfänglichen Gewichtungsvektoren zeigt auch die Epochenanzahl eine größere Variabilität. (Das Enkoder-Problem in Tabellen 1 und 2 weist deswegen ein anderes Abbruchkriterium $E_{\min}$ auf, damit der Backpropagation-Algorithmus eine Chance bekam, zu Ende zu rechnen.) □

Tabelle 3 Mittlere Epochenanzahl der asymptotisch stabilen Parameteradaption

Problem	$E_{\min}$	(2.3.3)	(2.4.10)
a) 3-2-4-Regression	10^0	$195 \pm 95\%$	$58 \pm 70\%$
b) 3-2-4-Approximation	10^{-4}	$1.070 \pm 140\%$	$189 \pm 115\%$
c) Quadratisch ($n = 76$)	10^{-16}	$464 \pm 17\%$	$118 \pm 9\%$
d) Potenz 1,8 ($n = 76$)	10^{-4}	$486 \pm 29\%$	$84 \pm 23\%$
e) Potenz 3,8 ($n = 76$)	10^{-16}	$28 \pm 10\%$	$37 \pm 14\%$
f) 8-3-8-Enkoder	10^{-4}	$1.380 \pm 60\%$	$300 \pm 60\%$

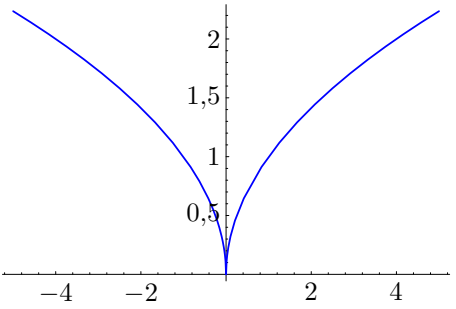


Abb. 32 Die nicht konvexe Fehlerfunktion $w \mapsto \sqrt{|w|}$

Grenzen dieser Verfahren (2.5.6) Die Verfahren des Abschnitts 2.4 sind alle für strikt konvexe Fehlerfunktionen zugeschnitten. Dies gilt auch für die Parameteradaption. Ein kritisches Beispiel stellt die Fehlerfunktion (2.5.1;E) mit $p = 1/2$ dar. Ihr Gradient ist singulär, wenn eine Koordinate verschwindet (insbesondere im globalen Minimum 0). Diese *Wurzelsingularitäten* sind für $n = 1$ in Abb. 32 und für $n = 2$ in Abb. 33 veranschaulicht.

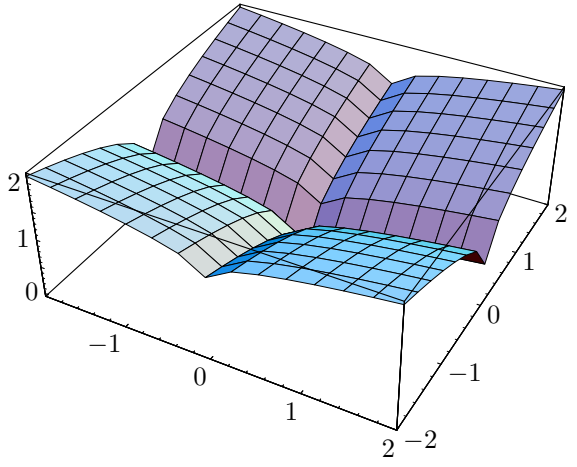


Abb. 33 Die 2D-Wurzelsingularität $w \mapsto \sqrt{|w_1|} + 2\sqrt{|w_2|}$

Die 2D-Wurzelsingularität zeigt zwei Probleme auf: Im Lauf des Lernens wächst der Betrag des Gradienten unbeschränkt, und der Winkel zwischen dem Gradienten und der Richtung zum Minimum nähert sich $\pi/2$.

(i) Das erste Teilproblem allein kann mit Normierung des Gradienten und Parameteradaption umgangen werden, wenn die Richtung des negativen

Gradienten zum Minimum zeigt. Das ist beispielsweise bei der eindimensionalen Wurzelsingularität $w \mapsto \sqrt{|w|}$ der Fall. Beim regulären Gradientenabstieg wächst der Betrag des Gradienten in der Nähe des Minimums, so daß jede gute Annäherung mit einem weiten Schritt weg vom Minimum bestraft wird (siehe Abb. 34). Dieses Verhalten kann übrigens auch jedes Parameteradaptionsverfahren zeigen, das nicht normierte Gradienten verwendet.

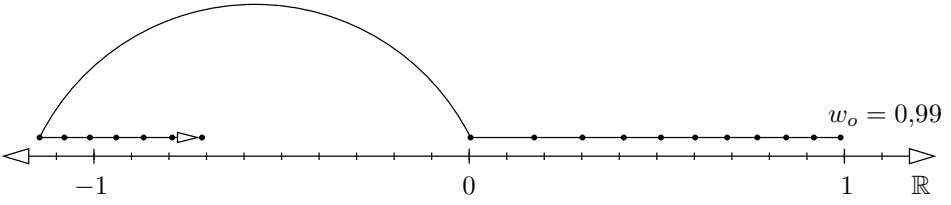


Abb. 34 Gradientenabstieg in der Wurzelsingularität $w \mapsto \sqrt{|w|}$ mit $\eta = 0{,}1$

Die naheliegende Idee, einen normierten Gradientenabstieg $w^{i+1} = w^i - \eta \nabla E_{w^i} / |\nabla E_{w^i}|$ zu benutzen, führt auf ein Abtastraster mit Schrittweite η. In unserem Beispiel muß es zu Zyklen kommen (siehe Abb. 35).

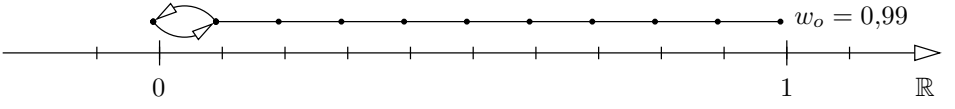


Abb. 35 Normierter Gradientenabstieg für $w \mapsto \sqrt{|w|}$ mit $\eta = 0{,}1$

Für die Parameteradaptionsverfahren zahlt es sich aus, daß die Schrittweite geometrisch verkleinert werden kann. So kann die eindimensionale Wurzelsingularität mit der asymptotisch stabilen Parameteradaption (2.3.3) unter Verwendung normierter Gradienten problemlos minimiert werden (siehe Abb. 36).

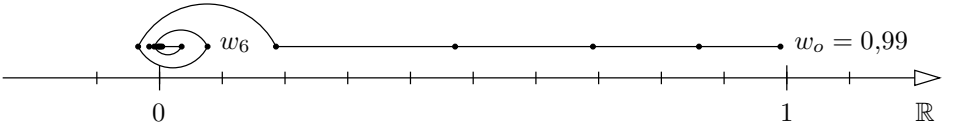


Abb. 36 Asymptotisch stabile Parameteradaption für $w \mapsto \sqrt{|w|}$ mit $\eta_o = 0{,}1$, $\zeta = 1{,}3$ und normiertem Gradienten

Ein ähnliches Ergebnis wird in ⟦Salomon und van Hemmen 1996⟧ für die Parameteradaption mit normiertem Gradienten (2.2.2;iii) erzielt, aber angesichts der folgenden Resultate etwas überbewertet.

(ii) Das zweite Teilproblem, nämlich daß die Richtung des Gradienten mehr und mehr senkrecht zur Richtung zum Minimum steht, erscheint fatal: alle Verfahren, die auf Gradientenbildung basieren, hoffen zumindest darauf, daß die Richtung des Gradienten ein hilfreicher Fingerzeig ist. In Abb. 37 wird demonstriert, wie die Parameteradaption mit normiertem Gradienten (2.2.2;iii) auf einer Koordinatenachse eingefangen wird. Die Lernrate zerfällt zu Null während der Gradient nahezu senkrecht auf eben jener Koordinatenachse – und damit auch auf der Richtung zum Minimum – steht. Die asymptotisch stabile Parameteradaption mit normierten Polak-Ribière-Richtungen zeigt das gleiche Phänomen. □

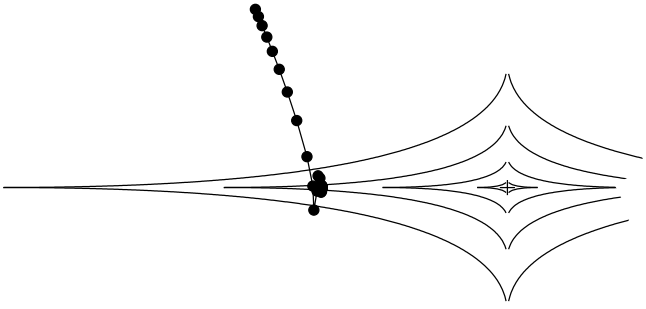


Abb. 37 2D-Wurzelsingularität: Parameteradaption mit normiertem Gradienten

Im allgemeinen kann auch – weit entfernt von einem typischen Minimum oder auch in der Nähe eines singulären Minimums – die asymptotisch stabile Parameteradaption instabil werden. Das motiviert, die in Bemerkung (2.3.4;iii) geäußerte Idee zur Definition einer stabilen Parameteradaption umzusetzen.

Stabile Parameteradaption (2.5.7) Sei $\zeta > 1$ beliebig, aber fest, und $\eta_o > 0$ beliebig. Wähle im t-ten Schritt des Verfahrens einen Vektor $g^t \in \mathbb{R}^E$, der den Bedingungen von (2.3.3) genügt. Sei $e\colon \eta \mapsto E(w^t - \eta g^t)$ eine durch E, g^t und w^t gegebene, eindimensionale Fehlerfunktion. Wiederhole (bis der Gradient verschwindet, ein oberes Limit für t oder ein unteres

Limit $E_{\min}$ für den Fehler erreicht ist) die Iterationsvorschrift $w^{t+1} = w^t - \eta_{t+1} g^t$ mit

$$\eta_{t+1} = \begin{cases} \dfrac{\eta_t \zeta / 2}{1 + \dfrac{e(\eta_t \zeta) - e(0)}{\eta_t \zeta g^t \nabla E(w^t)}} & \text{für } e(0) < e(\eta_t \zeta) \\ \eta_t / \zeta & \text{falls } e(\eta_t/\zeta) \leq e(\eta_t \zeta) \leq e(0) \\ \eta_t \zeta & \text{sonst,} \end{cases}$$

falls sich eine Verkleinerung des Fehlers ergibt, d. h. $E(w^{t+1}) < E(w^t)$. Anderenfalls setze $g^t = \nabla E(w^t)$ bzw. $g^t = \nabla E(w^t)/|\nabla E(w^t)|$ und $\eta_{t+1} = \eta_t/\zeta^k$ mit dem kleinsten $k \in \mathbb{N}$, so daß der Fehler sich verkleinert. □

Bemerkung (2.5.8) Diese Variante sollte bei konvexen Fehlerfunktionen etwas langsamer sein als die asymptotisch stabile Parameteradaption (2.3.3), da durch zusätzlichen Rechenaufwand in jedem Schritt – und nicht nur in der Nähe eines konvexen Minimums – eine Verkleinerung des Fehler garantiert wird. Bei problematischeren Fehlerfunktionen kann sich ein Vorteil ergeben: Im Beispiel von Abb. 36 entfernt sich das Gewicht w_6 des sechsten Schritts der asymptotisch stabilen Parameteradaption wieder leicht vom Minimum; bei der stabilen Parameteradaption kann das nicht der Fall sein, wie in Abb. 38 zu sehen ist.

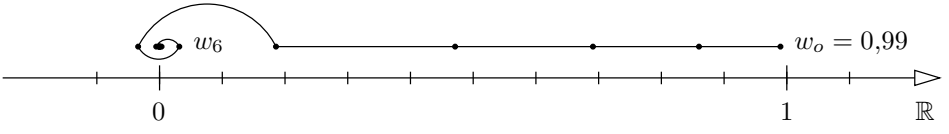


Abb. 38 Stabile Parameteradaption für $w \mapsto \sqrt{|w|}$ mit $\eta_o = 0{,}1$, $\zeta = 1{,}3$ und normiertem Gradienten

So ein schwieriges Probleme wie die mehrdimensionale Wurzelsingularität kann damit aber auch nicht gelöst werden, da auch hier im Verlauf des Lernens der Gradient nahezu senkrecht zur Richtung zum Minimum stehen wird. □

2.6 Online-Verfahren

Definition (2.6.1) Der *Online-Backpropagation-Algorithmus* ist eine Variante des Backpropagation-Algorithmus, bei der die Gewichtungsaktualisierung schon nach dem Präsentieren eines jeden Eingabe-Ausgabe-Musters erfolgt. Die Variable v des Backpropagation-Algorithmus (1.3.3) wird nicht benötigt; die Änderung von w im Schritt (iii) entfällt und wird dafür innerhalb der Schleife über die Muster im Schritt (ii) anstelle der Änderung von v durchgeführt. Im Unterschied zum Online-Backpropagation-Algorithmus heißt der reguläre Backpropagation-Algorithmus auch *Batch-Backpropagation-Algorithmus*; man spricht auch vom *Online-Modus* oder *Batch-Modus*. □

Bemerkung (2.6.2) Der Online-Modus scheint notwendig, wenn nicht alle Trainingsmuster zu Beginn des Lernens bekannt sind, der Batch-Modus zu zeitaufwendig wäre oder der Trainingsdatensatz zu redundant ist. □

Präsentationsreihenfolge (2.6.3) Im Online-Modus lautet die Lernregel

$$\Delta w^i = -\eta \nabla E^i|_{w^{i-1}} \quad \text{mit} \quad E^i = w \mapsto \mathrm{err}_w(p^i, t^i),$$

wobei (p^i, t^i) das i-te Eingabe-Ausgabe-Muster ist. Es erfolgt also ein sukzessiver Gradientenabstieg in *verschiedenen* Fehlerfunktionen, nämlich den Einzelfehlern der Trainingsmuster. Die Trajektorie $i \mapsto w^i$ der Gewichtungsvektoren hängt wesentlich von der Präsentationsreihenfolge $i \mapsto (p^i, t^i)$ der Muster ab.

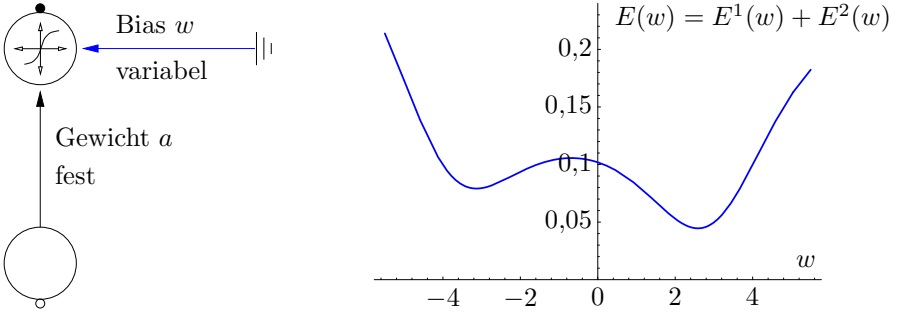


Abb. 39 Einfaches Netz mit Fehlerfunktion für zwei Muster

Sei beispielsweise das einfache „Netz“ aus Abb. 39 mit einem eingefrorenen Gewicht a und nur einem variablen Gewicht w gegeben. Bei zwei Mustern kann die Gesamtfehlerfunktion wie in Abb. 39 aussehen; sie setzt sich aus zwei Einzelfehlerfunktion, wie in Abb. 40 gezeigt, zusammen. Zwar kann jeder Einzelfehler verschwinden, nicht aber der Gesamtfehler; das hängt damit zusammen, daß die Bedingung $t^i = \sigma(ap^i + w)$ i. allg. nicht konsistent für $i = 1$ und $i = 2$ erfüllt werden kann.

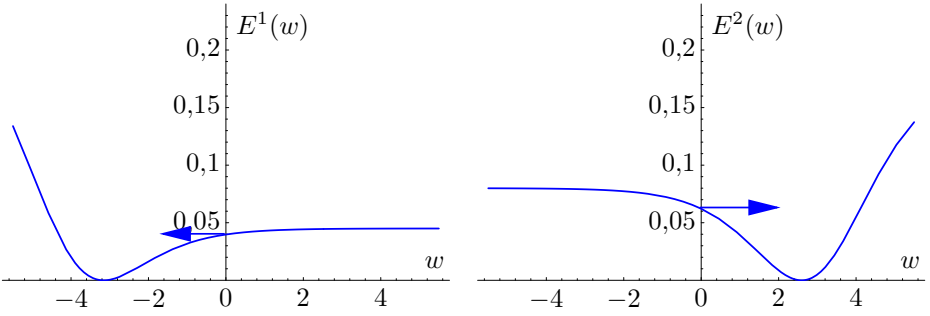


Abb. 40 Einzelfehlerfunktionen; die Gradienten dieser Funktionen an einem bestimmten Punkt haben nichts miteinander zu tun

Werden die Muster periodisch angelegt, $(p^i, t^i) = (p^{i+|T|}, t^{i+|T|})$, dann stellt obige Lernregel ein dynamisches System dar, das in diskreten Schritten von einer periodischen Kraft $-\nabla E^i$ angetrieben wird. Solche Systeme *können* chaotisches Verhalten zeigen, das dazu führen *kann*, daß durch periodische Präsentierung von Mustern aus lokalen Minima herausgefunden wird. Aber selbst im obigen einfachen Beispiel zeigen Experimente ⟦Radons et al. 1990⟧, daß die Attraktoren des Systems, gegeben eine bestimmte Lernrate, weder das globale Minimum enthalten müssen noch durch Mittelung nahe an das globale Minimum kommen.

Eine *zufällige Präsentationsreihenfolge* erzeugt da schon ein sehr viel besseres Systemverhalten: $-\nabla E^i$ ist dann eine stochastische Kraft. Angenommen, das Problem ist *perfekt erlernbar*, d. h. es gibt ein lokales Minimum w^* mit $E(w^*) = 0$, das zugleich ein globales Minimum ist: Eine weiteres lokales Minimum w^l mit $E(w^l) > 0$ erfährt dann (bei geeigneten Regularitätsvoraussetzungen) für einige Muster eine Kraft $-\nabla E^i \neq 0$. Bei

genügend hoher Lernrate und – durch Zufall bedingt – geeignetem Zusammenspiel von solchen Mustern kann der Bereich des lokalen Minimums w^l verlassen werden. Das bedeutet aber nicht notwendigerweise, daß der Bereich des globalen Minimums gefunden werden muß. Ist jedoch einmal das globale Minimum erreicht, verschwinden alle Einzelfehler und damit auch alle Kräfte.

Ist andererseits das Problem nicht perfekt erlernbar, dann kann es sein, daß bei fester Lernrate η das Lernverfahren nicht konvergiert. Die wesentliche Idee ist dann, η im Lauf des Lernens zu verkleinern. Eine große Lernrate zu Beginn des Lernprozesses bewirkt, daß ungeeignete lokale Minima leicht verlassen werden können; eine zu Null zerfallende Lernrate bringt dann das Verfahren hoffentlich bei einem globalen Minimum zum Stillstand.

Hier drängt sich die Analogie zu Simulated annealing (siehe Unterabschnitt (5.2.6)) auf; dort spielt statt η die Temperatur die Rolle des Rauschparameters. Es ist also zu erwarten, daß unter der Dynamik des Simulated annealing die Verteilung der w^i asymptotisch einer Gibbs-Verteilung folgt. Dann muß es aber auch zu jedem Problem ein geeignetes Abkühlungsschema geben, so daß eine *globale Optimierung* erreicht wird. Eine weitgehende Analyse dieses Problems wird in [[Heskes und Kappen 1993]] gegeben. □

2.7 Wertung

Das hier vorgestellte Trade-off-Theorem zeigt, daß die beim Backpropagation-Algorithmus verwendete Methode des Gradientenabstiegs denkbar ungeeignet ist (davon bleibt jedoch die in Kapitel 1 studierte und auch im Backpropagation-Algorithmus verankerte sehr effiziente Gradientenberechnung unberührt). Daher wurde in diesem Kapitel eine neue Optimierungsmethode entwickelt und im Zusammenhang mit Standardmethoden der konvexen Optimierung vorgestellt. Aus vergleichenden Studien ergibt sich immer wieder, daß Lernverfahren sich in manchen Domänen bewähren, in denen andere versagen oder ineffizient sind, und umgekehrt für andere Anwendungsgebiete. Den „besten“ Lernalgorithmus scheint es nicht zu geben, denn das nichtlineare Optimierungsproblem ist i. allg. zu komplex.

Für Lernregeln in neuronalen Netzen sind Geschwindigkeit und Stabilität vorrangige Beurteilungskriterien. Die in dieser Hinsicht in diesem Kapitel aus einem bekannten Verfahren entwickelte asymptotisch stabile Parameteradaption vereint einige Vorzüge: Es gibt für sie einen weitreichenden mathematischen Konvergenzbeweis. Sie benutzt nur relativ einfach zu berechnende Größen wie den Gradient der Fehlerfunktion und die Fehlerfunktion selbst. Sie erfordert kein kritisches Einstellen neuer Parameter. Sie erlaubt die Einbeziehung von Zweiter-Ordnung-Information (z. B. durch auch vom Konvergenzbeweis abgedeckte Polak-Ribière-Richtungen) ohne aufwendige Liniensuche. Die Eigenschaft der *asymptotischen* Stabilität ist ein ausgewogener Kompromiß zwischen der stärkeren Forderung der Stabilität, deren Erfüllung viel Rechenzeit ohne sichtbaren Gewinn kostet, und potentieller Instabilität, die konvergenzzerstörend wirken kann.

Die asymptotisch stabile Parameteradaption ist für den Batch-Modus gemacht. Trotzdem kann sie erfolgreich in einem „halben“ Online-Modus betrieben werden: dabei werden jeweils so viele Muster zusammengefaßt, daß sich die jeweiligen Fehlerfunktionen ähneln. Durch die Robustheit der Parameteradaption ist hier ein sinnvoller Kompromiß möglich, wenn so viele Muster existieren, daß Batch-Lernen zu zeitaufwendig wäre.

3 Neuronale Netze als Regressionsmodell

Dieser Trick, sich gleichsam an den eigenen (mathematischen) Haaren aus dem (statistischen) Sumpf zu ziehen, könnte mithin als Münchhausen-Verfahren bezeichnet werden (im Amerikanischen wird es entsprechend bootstrap procedure genannt [...]).

— Spektrum der Wissenschaft, Juli 1983

Statistik und neuronale Netze haben viel gemeinsam, vor allem wenn das Ziel ist, die Struktur verrauschter Daten zu verstehen. In diesem Kapitel wird das Lernen neuronaler Netze als statistische Regression aufgefaßt ⟦siehe auch White 1989a, White 1989b und Paaß 1994⟧ und damit als Schätzproblem formuliert. Verschiedene statistische Methoden erlauben die Abschätzung der Güte des Lernergebnisses, und zwei davon (Delta-Methode und Bootstrap) werden detailliert vorgestellt. Die Gütebeurteilungen unterscheiden sich signifikant, was durch spezielle Probleme neuronalen Lernens verursacht wird. Eine Lösung der Probleme wird im darauffolgenden Kapitel entwickelt.

3.1 Lernen in einem statistischen Kontext

Die Zufälligkeit der Daten und Regression (3.1.1) In vielen Anwendungsbereichen können Daten nicht ohne Fehler erhoben werden. Manchmal, etwa bei physikalischen Experimenten, sind diese Fehler nicht systematischer, sondern zufälliger Natur, und es kann ein explizites Wahrscheinlichkeitsmodell der zufälligen Einflüsse angegeben werden. So ein Modell kann aus theoretischen Überlegungen heraus (z. B. über den zentralen Grenzwertsatz der Wahrscheinlichkeitstheorie) oder durch empirische Versuche entstehen. Die Pattern-Target-Relation besteht dann aus Realisierungen p^i und t^i der Zufallsvariablen P^i und T^i, die eine gemeinsame Verteilung besitzen.

Der funktionale Zusammenhang zwischen Zufallsvariablen wird i. allg. *Regression* genannt. Die Abbildung $p^i \mapsto \mathrm{E}(T^i|P^i = p^i)$ heißt *Regressionslinie*; dabei ist $\mathrm{E}(T^i|P^i = p^i)$ die bedingte Erwartung von T^i, falls P^i den Wert p^i angenommen hat. In der statistischen *Regressionsanalyse* wird nun versucht, die Regressionslinie anzunähern. Kennt man nun das Modell für die Verteilung von T^i gegeben P^i und das für die Verteilung von P^i, dann kann damit ja wieder die gemeinsame Verteilung von P^i und T^i berechnet werden. Von Interesse ist hierbei das Auffinden der Regressionslinie. Da dort die Zufallsvariable P^i aber nur als Bedingung auftritt, kann sie bei der zukünftigen Behandlung als deterministisch betrachtet werden. In der weiteren Darstellung wird zwischen Zufallsvariable und Realisierung nicht mehr durch Groß- und Kleinschreibung unterschieden. Auch ist die folgende Darstellung sehr knapp gehalten; für eine weitergehende Darstellung siehe ⟦Kockelkorn 1997⟧. □

Ein Regressionsmodell (3.1.2) In einem *Regressionsmodell* ist ein Zusammenhang zwischen Eingabe-Ausgabe-Mustern zu finden. Dabei wird zunächst angenommen, daß die Muster durch einen Zusammenhang

$$t^i = \mathrm{out}_{w^\star}(p^i) + \epsilon^i$$

modelliert werden können. $w^\star$ sei der Gewichtungsvektor eines „wahren" Modells der Daten durch ein neuronales Netz, wie z. B. in Abb. 41. Die ϵ^i seinen unabhängige, identisch verteilte Zufallsvariablen mit Erwartungswert Null (z. B. *Meßfehler*). Die *Trainingsmultimenge* T wird dann durch eine Multimenge n solcher Datenpaare (p^1, t^1), …, (p^n, t^n) gegeben. Da i. allg. bei neuronalen Netzen die Abbildung $w \mapsto \mathrm{out}_w$ nichtlinear ist, wird auch von *nichtlinearer Regression* gesprochen. □

Likelihood (3.1.3) Durch ϵ^i ist auch $t^i = \mathrm{out}_{w^\star}(p^i) + \epsilon^i$ eine Zufallsvariable, daher auch (p^i, t^i) und somit auch die Trainingsmultimenge T. Die Wahrscheinlichkeitsverteilung für ϵ^i induziert eine Wahrscheinlichkeitsverteilung für T, die wegen der Unabhängigkeit der ϵ^i faktorisiert: Habe das Eingabe-Ausgabe-Muster (p^i, t^i) eine Wahrscheinlichkeitsverteilung mit Dichte q_i, so ist

$$L_T(w) := \prod_i q_i(p^i, t^i)$$

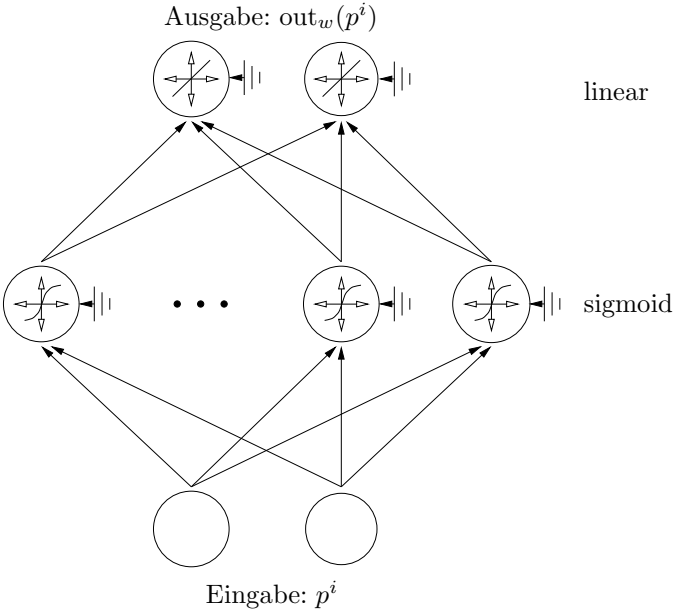


Abb. 41 Das angenommene wahre Modell der Daten

die Dichte der Wahrscheinlichkeitsverteilung von T.

Die Abbildung $w \mapsto L_T(w)$ heißt *Likelihood*. Sie ist keine Dichte, da das Integral über den Gewichtungsraum einen beliebigen positiven Wert ergeben kann; insbesondere wurde dem Gewichtungsraum auch keine Wahrscheinlichkeitsstruktur aufgeprägt.

In der Statistik heißt jede meßbare Abbildung, die jeder Trainingsmultimenge T genau einen Gewichtungsvektor zuordnet, ein *Schätzer* des Gewichtungsvektors. Jeder Parameter $\hat{w}_n$ mit

$$\hat{w}_n := \operatorname*{argsup}_w L_T(w)$$

heißt *Maximum-Likelihood-Schätzer* von w. Ist g eine Abbildung vom Gewichtungsraum nach Y, dann heißt jede Abbildung, die Trainingsmultimengen ein Element von Y zuordnet ein Schätzer von $g(w)$. □

Bemerkungen (3.1.4) (i) An einen Schätzer ist also – abgesehen vom richtigen Definitions- und Wertebereich – keine mathematische Bedingung geknüpft. Im Prinzip verkörpert eine spezielle Trainingsmultimenge T eine

Realisierung der entsprechenden Zufallsvariable; des weiteren ist jeder Gewichtungsvektor w kompatibel mit einem beobachteten T: selbst bei beliebig kleiner Likelihood von w scheidet w nicht von vornherein aus.

(ii) Jedoch ist es viel plausibler, daß ein bestimmtes beobachtetes T von einem Modell mit großer Likelihood erzeugt wurden als von einem Modell mit kleiner Likelihood (daher der Name). Unter sehr allgemeinen Bedingungen und im Fall, daß das Modell richtig ist, hat ein Maximum-Likelihood-Schätzer asymptotisch (bezüglich n) sehr wünschenswerte Eigenschaften: $\hat{w}_n$ ist ⟦White 1992⟧

- *konsistent*, d. h. $\hat{w}_n$ konvergiert in Wahrscheinlichkeit gegen $w^\star$, d. h.

$$\text{für alle } \varepsilon > 0 \text{ ist } \lim_{n\to\infty} P(|\hat{w}_n - w^\star| > \varepsilon) = 0,$$

- *asymptotisch erwartungstreu* (asymptotisch gilt $\mathrm{E}(\hat{w}_n) = w^\star$),
- *asymptotisch effizient*, d. i. ein asymptotisch erwartungstreuer Schätzer mit minimaler Varianz,
- *asymptotisch normalverteilt*, d.h. konvergiert in Verteilung gegen eine Normalverteilung, und
- *äquivalent zum Kleinste-Quadrate-Schätzer falls* ϵ_i normalverteilt ist.

Obige Aussagen (wie auch einige folgende) müssen präzisiert werden, weil es eine Reihe äquivalenter Gewichtungsvektoren mit gleicher Netzfunktion $\mathrm{out}_{w^\star}$ gibt ($w \mapsto \mathrm{out}_w$ ist in der Regel nicht injektiv). Dieses sog. *Identifizierungsproblem* wird in Kapitel 4 behandelt, wo auch ein effektiver Gewichtungsraum mit einer geeigneten Metrik vorgestellt wird. Der effektive Gewichtungsraum entsteht durch Identifizierung äquivalenter Gewichtungsvektoren; er ersetzt den Gewichtungsraum $\mathbb{R}^E$.

(iii) Abb. 42 ist ein illustratives Beispiel für das Prinzip der Maximum-Likelihood-Schätzung. Es soll die Erfolgswahrscheinlichkeit w bei n unabhängigen *Bernoulli-Experimenten* (das sind Experimente mit binärem Ausgang; einer der beiden Ausgänge – Erfolg genannt – hat Wahrscheinlichkeit w) geschätzt werden. Die Anzahl k der Erfolge gehorcht der Binomialverteilung, hat also die Wahrscheinlichkeit

$$L_k(w) = \binom{n}{k} w^k (1-w)^{n-k}.$$

Die Likelihood $w \mapsto L_k(w)$ ist in Abb. 42 für verschiedene k gezeichnet. Das Maximum der Likelihood ist je nach beobachtetem k verschieden: nach kurzer Rechnung ergibt sich $k \mapsto k/n$ als Maximum-Likelihood-Schätzer. □

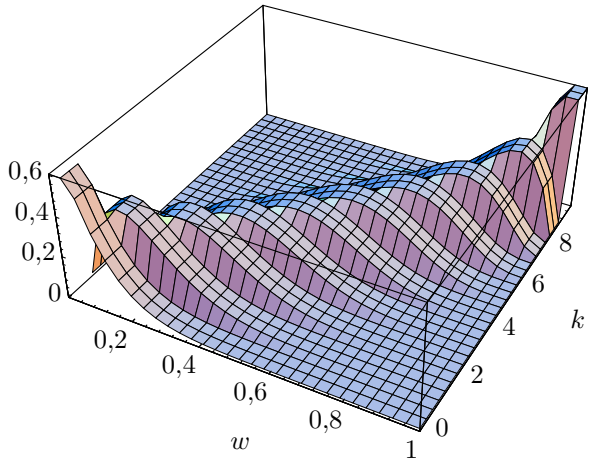


Abb. 42 Die Likelihood $w \mapsto L_k(w)$ aus (3.1.4;iii) mit $n = 9$

Annahmen über die Verteilung der Beispiele (3.1.5) Die Behauptung in (3.1.4;ii), daß der Maximum-Likelihood-Schätzer $\hat{w}_n$ mit dem Kleinste-Quadrate-Schätzer übereinstimmt, falls der Fehler ϵ^i normalerteilt ist, läßt sich sofort anhand der Likelihood nachprüfen. Der *Kleinste-Quadrate-Schätzer* ist durch Minimierung von $w \mapsto \sum_i (t^i - \mathrm{out}_w(p^i))^2$ gegeben, ist also schon als Gesamtfehler (1.2.8) bekannt. In diesem Sinn stellt die Maximierung der Likelihood eine Verallgemeinerung dar.

Andererseits zeigt die Betrachtung auch, daß – entgegen der landläufigen Meinung vieler Experten für neuronale Netze – eben doch durch die Auswahl der Kostenfunktion implizit Annahmen über die Verteilung der Beispiele gemacht werden. Zum Beispiel ist bei Bernoulli-verteiltem Fehler der Maximum-Likelihood-Schätzer äquivalent zur Minimierung der Kostenfunktion der relativen Entropie. Wenn die Beispiele als mit Zufall behaftet gelten, dann ist die Wahl der Kostenfunktion also auch weniger frei, als der Abschnitt 1.5 glauben macht. □

3.2 Beurteilung von Schätzern

Konsistenz ist eine Art Minimalforderung an einen sinnvollen Schätzer: Bei steigenden Datenmengen soll der geschätzte Wert immer näher am wahren Wert liegen. Es folgen weitere Kriterien eines Schätzers.

Erwartungstreue, Bias und Risiko (3.2.1) Je nachdem, welcher wahre Gewichtungsvektor w^* in einem neuronalen Netz die Daten erzeugt hat, hat die Trainingsmultimenge eine andere Dichte $T \mapsto L_T(w^*)$. $\mathrm{E}_w(G)$ sei der Erwartungswert von G bezüglich der Dichte $T \mapsto L_T(w)$, falls dieser existiert (und insbesondere G meßbar, also eine Zufallsvariable ist). Ein Schätzer G von $g(w)$ heißt *erwartungstreu*, falls für alle w gilt $\mathrm{E}_w(G) = g(w)$. Der Ausdruck

$$b(G, w) = \mathrm{E}_w(G) - g(w)$$

heißt *Verzerrung* oder auch *Bias* des Schätzers G von $g(w)$. Der Bias erwartungstreuer (engl. *unbiased*) Schätzer ist null. Das *Risiko* von G

$$R(G, w) := \mathrm{E}_w\left(\big(G - g(w)\big)^2\right)$$

ist als Erwartungswert einer *Verlustfunktion*, hier $(x, w) \mapsto (G(x) - g(w))^2$, bezüglich der Verteilung mit Dichte $T \mapsto L_T(w)$ definiert; es hängt noch vom Parameter w ab.

Das Risiko eines Schätzers ist die Summe seiner Varianz und Quadrat seines Bias, denn

$$\begin{aligned} R(G, w) &= \mathrm{E}_w\left(\big(G - \mathrm{E}_w(G) + \mathrm{E}_w(G) - g(w)\big)^2\right) \\ &= \mathrm{E}_w\left(\big(G - \mathrm{E}_w(G)\big)^2\right) + \mathrm{E}_w\left(\big(g(w) - \mathrm{E}_w(G)\big)^2\right) \\ &= \mathrm{Var}_w(G) + (b(G, w))^2. \end{aligned}$$

Diese Beobachtung legt nahe, von zwei erwartungstreuen Schätzern den mit kleinerer Varianz zu bevorzugen, denn er hat kleineres Risiko. □

Konfidenzbereiche (3.2.2) Sei P_w das zur Dichte $L_\bullet(w)$ gehörige Wahrscheinlichkeitsmaß, also

$$P_w(A) := \int_A L_T(w)\, dT.$$

Sei $C(T)$ für jede mögliche Trainingsmultimenge T eine abgeschlossene Teilmenge des Wertebereichs von g. Falls für ein bestimmtes $0 \leq \gamma \leq 1$ und für alle w gilt

$$P_w\big(\{T \mid g(w) \in C(T)\}\big) \geq \gamma,$$

so heißt $C(T)$ *Konfidenzbereich von* $g(w)$ *zum Konfidenzniveau* γ *zur Beobachtung* T (siehe z. B. Abb. 43). Die natürliche Interpretation ist die, daß bei unbekanntem Gewichtungsvektor $w^\star$ des wahren Modells mit Wahrscheinlichkeit von mindestens γ eine solche Trainingsmultimenge beobachtet wird, so daß der zu schätzende Wert $g(w^\star)$ in $C(T)$ liegt. □

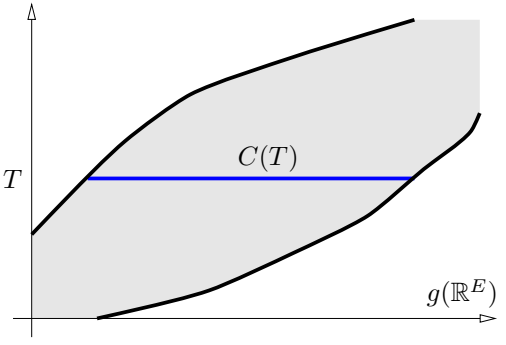


Abb. 43 Konfidenzbereich $C(T)$ der zufälligen Trainingsmultimenge T

Angabe von Konfidenzbereichen (3.2.3) Besonders wertvoll sind Konfidenzbereiche dann, wenn sie möglichst kleine (und zusammenhängende) Bereiche bei einem möglichst hohen Niveau sind. Eine Möglichkeit zur Konstruktion eines Konfidenzbereichs ist es, für einen Schätzer G von $g(w)$ mit Hilfe der Dichten $T \mapsto L_T(w)$ Zahlen $d(T)$ so zu bestimmen, daß für alle w gilt

$$P_w\big(\{T \mid |g(w) - G(T)| \leq d(T)\}\big) \geq \gamma.$$

Dann ist $C(T) = [G(T) - d(T), G(T) + d(T)]$ ein Konfidenzbereich. Bei erwartungstreuen Schätzern G kann $d(T)$ als ein Vielfaches eines Schätzers der Standardabweichung von $G(T)$ angesetzt werden.

Schätzer hängen von der Zufallsvariable T ab, sind daher selbst Zufallsvariablen und besitzen demzufolge eine Verteilung. Bei erwartungstreuen

Schätzern G von $g(w)$, deren Verteilung bekannt ist, kann ein Konfidenzbereich als Intervall mit den $(1-\gamma)/2$- und $(1+\gamma)/2$-Quantilen der Verteilung angegeben werden (siehe Abb. 44). Im allgemeinen ist die Konstruktion von Konfidenzbereichen jedoch nicht trivial. □

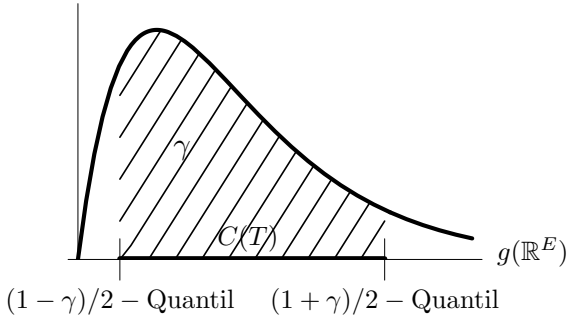


Abb. 44 Konfidenzbereich durch Quantile der Verteilung des Schätzers

3.3 Die Delta-Methode

Variabilität der Netzfunktion (3.3.1) Die Variabilität der Netzfunktion einer Maximum-Likelihood-Schätzung mit z. B. n Datenpaaren kann im Prinzip wie folgt bestimmt werden: Es werden immer wieder neue Datenmengen mit n Elementen beschafft, z. B. durch erneute mit Meßfehlern behaftete Messungen; mit diesen Datenmengen wird je eine neue Maximum-Likelihood-Schätzung durchgeführt. Die Variabilität der Netzfunktionen kann an jeder Stelle p der Eingabe (nicht nur bei den Beispielsdaten) als empirische Standardabweichung der vielen verschiedenen Ausgaben der Netzfunktionen bei p ermittelt werden. Dieses Vorgehen ist in Abb. 45 und 46 veranschaulicht. □

Vereinbarung (3.3.2) In den nachfolgenden Bildern dieses Kapitels ist die Netzfunktion eines wahren Zusammenhangs immer gestrichelt eingezeichnet; nach (3.1.2) mit normalverteiltem Fehler ϵ^i erzeugte Datenpaare sind als dicke Punkte markiert. □

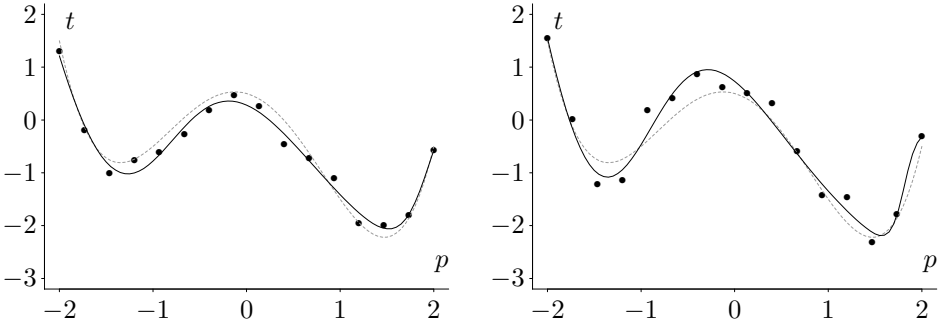


Abb. 45 Netzfunktionen $\text{out}_{\hat{w}}$ zweier Maximum-Likelihood-Schätzungen $\hat{w}$ mit verschiedenen Datenmengen des gleichen wahren Modells

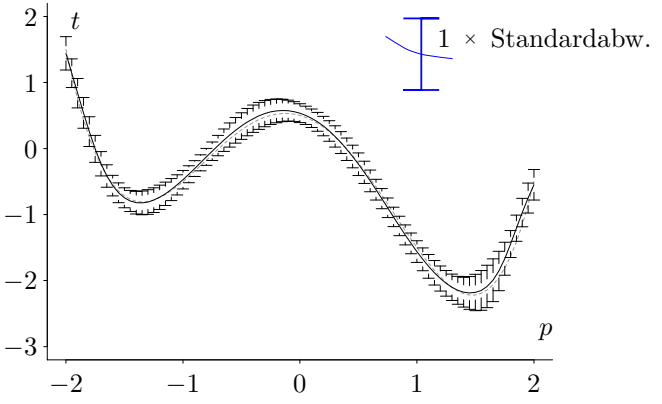


Abb. 46 Variabilität der Netzfunktion von Maximum-Likelihood-Schätzungen aus Abb. 45 mit je 16 Datenpunkten; die Fehlerbalken geben die empirische Standardabweichung von 20 Schätzungen mit je anderen Datensätzen an und die durchgezogene Linie ist die mittlere Netzfunktion

Solch ein Vorgehen wäre aber klarerweise eine Verschwendung von Ressourcen. Viel geschickter scheint es, die asymptotische Normalität der Likelihood auszunutzen:

Variabilität der Gewichtungsvektoren (3.3.3) Durch

$$\hat{\Sigma}(\hat{w}_n) = -(\mathrm{D}\,\mathrm{D}\log L_T|_{\hat{w}_n})^{-1}$$

wird eine mögliche Schätzung $\hat{\Sigma}(\hat{w}_n)$ der Varianz-Kovarianzmatrix des Maximum-Likelihood-Gewichtungsvektors gegeben. Falls die Likelihood

selbst schon normalverteilt ist, ergibt sich die Schätzung ganz offensichtlich; für eine (noch) nicht ganz normalverteilte Likelihood ist hofft man, daß $\hat{\Sigma}(\hat{w}_n)$ nicht weit von der wahren Varianz-Kovarianzmatrix entfernt ist. □

Bemerkungen (3.3.4) (i) Die negative Hessesche Matrix der sog. *Log-Likelihood* $\log \circ L_T$ heißt auch *Informationsmatrix*. Demnach ist $\hat{\Sigma}(\hat{w}_n)$ die inverse beobachtete Informationsmatrix.

(ii) Sind die Meßfehler ϵ^i normalverteilt mit Standardabweichung σ, dann ist die Log-Likelihood durch

$$\log(L_T(w)) = -\frac{1}{2\sigma^2}\sum_{i=1}^{n}(t^i - \mathrm{out}_w(p^i))^2 - \frac{n}{2}\log(2\pi\sigma^2)$$

gegeben, also i. w. durch die quadratische Fehlerfunktion des Netzes. Es können dann sehr leicht die in (2.4.1) besprochenen Algorithmen zur Berechnung der Hesseschen Matrix der Fehlerfunktion eingesetzt werden. Selbst für nicht-normalverteiltes ϵ^i sind in der Regel nur kleine Änderungen an den betreffenden Algorithmen bezüglich der Kostenfunktion nötig. Ist σ nicht bekannt, kann es durch die empirische Standardabweichung geschätzt werden. □

Delta-Methode (3.3.5) Mit Hilfe der *Delta-Methode* kann die Variabilität der Gewichtungsvektoren (3.3.3) in eine Variabilität der Netzfunktion (3.3.1) transformiert werden: Die Netzfunktion out_w wird dabei an einer festgehaltenen Stelle p betrachtet, dann die Abbildung $w \mapsto \mathrm{out}_w$ an der Stelle $\hat{w}_n$ in eine Taylorreihe entwickelt und nach dem ersten Glied abgebrochen. Die so entstehende Approximation ist eine affine Abbildung, die eine (asymptotisch) normalverteilte Zufallsvariable im Gewichtungsraum in eine (asymptotisch) normalverteilte Zufallsvariable im Raum $\mathbb{R}^O$ der Netzausgaben mit Erwartungswert $\mathrm{out}_{\hat{w}_n}(p)$ und Varianz-Kovarianzmatrix

$$\nabla_w \mathrm{out}_w(p)|_{\hat{w}_n}^T \cdot \hat{\Sigma}(\hat{w}_n) \cdot \nabla_w \mathrm{out}_w(p)|_{\hat{w}_n}$$

überführt. Abb. 47 zeigt diese Transformation graphisch. □

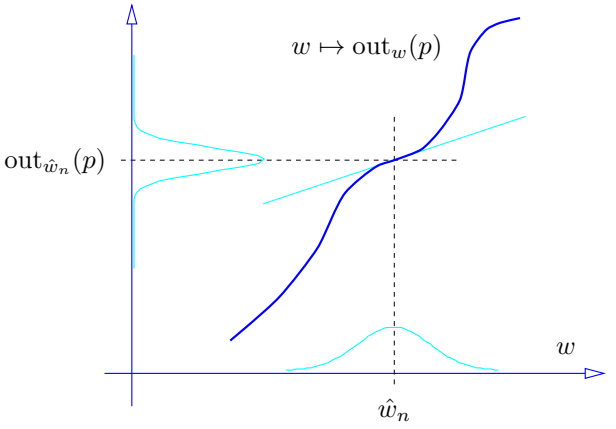


Abb. 47 Die Delta-Methode

Beispiel (3.3.6) Es seien 16 verrauschte Datenpaare eines wahren Zusammenhangs gegeben. In Abb. 48 ist die Netzfunktion der Maximum-Likelihood-Schätzung zusammen mit der nach der Delta-Methode ermittelten Standardabweichung eingezeichnet. □

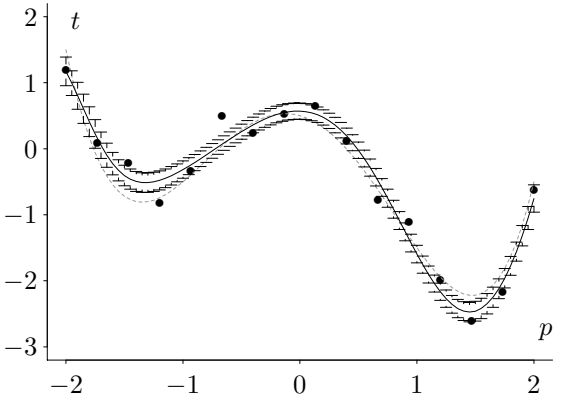


Abb. 48 Variabilität der Netzfunktion nach der Delta-Methode

Probleme der Delta-Methode (3.3.7) (i) Die so berechneten Standardabweichungen nutzen asymptotische Güteeigenschaften (3.1.4;ii) der Maximum-Likelihood-Schätzung aus. Deren Anwendbarkeit ist bei endlich vielen Datenpaaren aber nicht immer klar.

(ii) Die Delta-Methode selbst gibt sich mit einer ersten Näherung der Taylorentwicklung von $w \mapsto \mathrm{out}_w$ zufrieden; dies kann durch Einbeziehung höherer Glieder genauer gemacht werden: die Verteilung der Netzausgaben zu einer bestimmten Netzeingabe wird dann nicht mehr durch eine Normalverteilung approximiert.

(iii) In der Praxis ist das Auffinden der Maximum-Likelihood-Schätzung ein schwieriges Optimierungsproblem. Beispielsweise kann ein Gradientenverfahren in einem lokalen Maximum steckenbleiben (selbst wenn lokale suboptimale Maxima im Limes wegfallen, können sie gleichwohl bei endlich vielen Datenpaaren immer Probleme bereiten). In solch einem Fall ist nicht nur die Schätzung des Zusammenhangs suboptimal, sondern auch – was noch viel schlimmer ist – die Schätzung der Variabilität der Netzfunktion. Es kann, wie in Abb. 49 demonstriert, der falsche Eindruck einer recht wenig variablen Netzfunktion entstehen, die aber die tatsächlichen Fehler nicht widerspiegelt. Eine Analyse der *Residuen*, d. s. die Abweichungen $\mathrm{out}_w(p^i) - t^i$ der Daten (p^i, t^i) von der Netzfunktion, sollte so gut wie immer gemacht werden. Beispielsweise könnte mit Hilfe eines statistischen Tests geprüft werden, ob die Residuen so verteilt sind wie es das Modell für den Meßfehler ϵ^i vorsieht. □

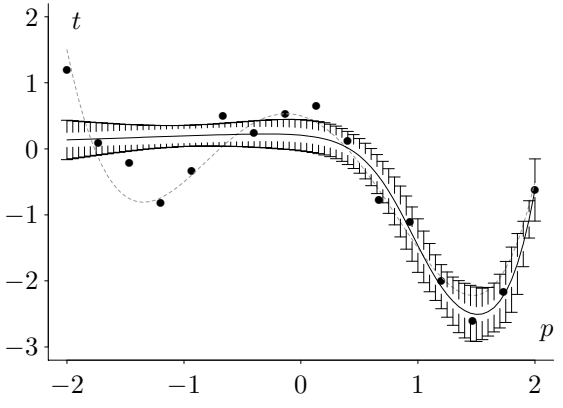


Abb. 49 Ein lokales Maximum der Likelihood ergibt weder eine gute Regression noch eine plausible Fehlerbeurteilung

3.4 Bootstrap

Bootstrap-Methode (3.4.1) Die *Bootstrap-Methode* ⟦Efron und Tibshirani 1993⟧ basiert auf der Verwendung von B Pseudo-Datenmultimengen $T^{*1}, \ldots, T^{*B}$, die jeweils durch n-maliges Ziehen mit Zurücklegen aus der ursprünglichen Datenmenge T erzeugt werden. Die Pseudo-Datenmultimengen werden auch *Bootstrap-Stichproben* genannt; sie haben genauso viele Elemente wie die Datenmenge T, aber manche Datenpaare aus T kommen gar nicht vor und manche mehrfach. Für jede dieser Bootstrap-Stichproben kann nun eine Maximum-Likelihood-Schätzung

$$\hat{w}^{*i} := \operatorname*{argsup}_w L_{T^{*i}}(w)$$

durchgeführt werden. Die Idee ist dann, daß bei gegebener Netzeingabe p die Variabilität von $\mathrm{out}_{w^{*i}}(p)$ als Approximation für die Variabilität der Netzfunktion betrachtet werden kann. Bei nur einem Ausgabeknoten kann z. B. die empirische Standardabweichung

$$\sqrt{\frac{1}{B-1}\sum_{i=1}^{B}(\mathrm{out}_{w^{*i}}(p) - \overline{\mathrm{out}}^B(p))^2} \qquad (s)$$

benutzt werden, wobei

$$\overline{\mathrm{out}}^B(p) := \frac{1}{B}\sum_{i=1}^{B}\mathrm{out}_{w^{*i}}(p)$$

die mittlere *Bootstrap-Regressionslinie* bedeute. □

Beispiel (3.4.2) Es sei der gleiche Datensatz wie für das Beispiel (3.3.6) gegeben. In Abb. 50 sind die einzelnen $B = 20$ Netzfunktionen der Bootstrap-Schätzungen gezeichnet. Durch Mittelung ergibt sich eine Bootstrap-Regressionslinie mit zugehöriger Standardabweichung (siehe Abb. 51). □

Probleme der Bootstrap-Methode (3.4.3) Durch die Verwendung von Maximum-Likelihood-Schätzungen ergeben sich in der Praxis auch die in (3.3.7;iii) geschilderten Probleme. In Abb. 50 sind offensichtlich einige Netzfunktionen durch Unzulänglichkeiten des Lernalgorithmus (z.B. eben

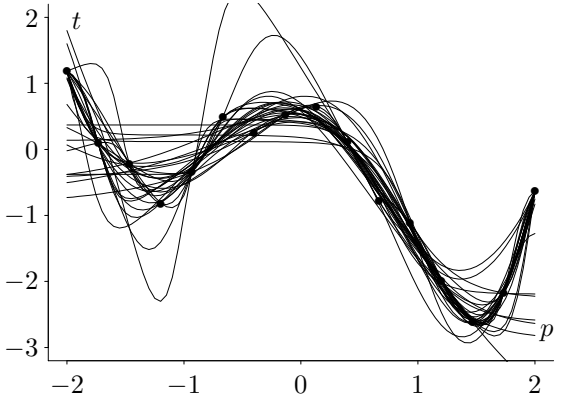


Abb. 50 Netzfunktionen von Bootstrap-Schätzungen

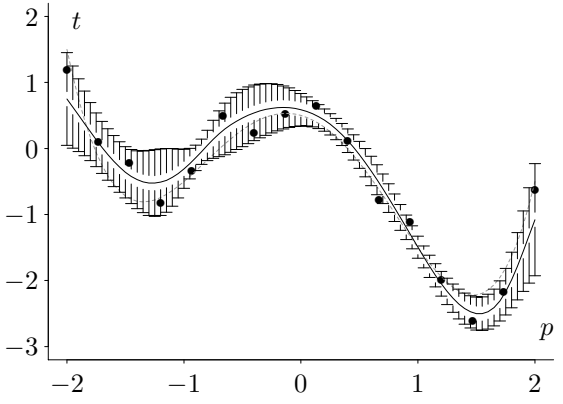


Abb. 51 Mittlere Bootstrap-Regressionslinie mit empirischer Standardabweichung

nicht das Maximum der Likelihood gefunden zu haben) entstanden. Solche Ergebnisse verschlechtern sowohl die Bootstrap-Regressionslinie als auch die Schätzung der Standardabweichung der Netzausgaben.

Tibshirani sieht dieses Problem als nützliche Eigenschaft; in einem Vergleich verschiedener Methoden zur Schätzung der Standardabweichung der Netzausgaben faßt er zusammen: „*We find that the bootstrap methods perform best, partly because they capture variability due to the choice of the starting weights*“ ⟦Tibshirani 1996⟧.

Dies drückt die Robustheit der Bootstrap-Methode zur Bestimmung der Standardabweichung bezüglich unzulänglicher Lernergebnisse aus; es bleibt

jedoch das Problem bestehen, daß unklar ist, *wovon* eigentlich die Standardabweichung approximiert wurde: (3.4.1;s) kann nicht die Standardabweichung der mittleren Bootstrap-Regressionslinie sein, denn dazu wäre eigentlich eine zweite Bootstrap-Ebene nötig („*Bootstrap the bootstrap!*“ ⟦Tibshirani 1995⟧); es kann auch nicht die Standardabweichung einer wirklichen Maximum-Likelihood-Schätzung sein, denn es sind unzulängliche Schätzungen verarbeitet. Aus dem gleichen Grund kann (3.4.1;s) auch nicht die Standardabweichung einer einzelnen durch lokale suboptimale Maxima der Likelihood verzerrten Schätzung sein.

In der Tat subsummiert (3.4.1;s) die Unzulänglichkeiten des Lernverfahrens, was im Fall einer zugrundeliegenden echten Maximum-Likelihood-Schätzung wie in Abb. 48 zu unnötig großen Fehlerbalken führt und im Fall einer zugrundeliegenden verzerrten (z.B. durch ein lokales Maximum der Likelihood zustandegekommenen) Schätzung zu einer ungenügenden Variabilitätsbeurteilung. □

3.5 Wertung und Ausblick

In diesem Kapitel wurde dargelegt, wie das Trainieren neuronaler Netze in natürlicher Weise als statistisches Schätzproblem aufgefaßt werden kann, wenn die zugrundeliegenden Daten zufälligen Einflüssen unterliegen. Insbesondere muß dann die Schätzung – d. h. das Ergebnis des Lernens, z. B. ein Gewichtungsvektor, eine Netzfunktion oder die Ausgabe des Netzes für eine bestimmte Eingabe – als eine mit Zufall behaftete Größe gesehen werden. Diese verallgemeinerte Sichtweise birgt die Möglichkeit in sich, in der Statistik etablierte Methoden zur Beurteilung der Güte einer Schätzung und damit der Güte des trainierten Netzes anzuwenden.

Die Standardabweichung eines erwartungstreuen Schätzers erweist sich als ein geeignetes Mittel zur Beurteilung, z. B. können auch Konfidenzbereiche damit bestimmt werden. In der Praxis zeigen verschiedene Methoden zur Bestimmung der Standardabweichung (beispielsweise der Netzausgabe) jedoch deutliche Unterschiede. Die Ursachen dafür liegen in Unzulänglichkeiten der verwendeten Lernverfahren. Nicht in jedem Versuchslauf wird das *globale* Optimum der Likelihood gefunden; manchmal bleiben

die Verfahren eben auch stecken. Dies kann – besonders im Falle von Bootstrap, das schon durch eine neue Bootstrap-Stichprobe die Likelihoodfunktion ändert – auch nicht immer den erreichten Likelihoodwerten angesehen werden.

In Abschnitt 4.7 wird eine neue Methode vorgestellt, die eine robuste Schätzung der Variabilität der Netzfunktion erlaubt, indem unzureichende Schätzungen mit Hilfe der *geometrischen Lageinformation* von Gewichtungsvektoren herausgefiltert werden. Zu diesem Zweck muß zunächst das Identifizierungsproblem gelöst werden und der Gewichtungsraum mit einer geeigneten Metrik versehen werden. Jedoch wird sich diese Vorgehensweise als so fruchtbar erweisen, daß viele andere Probleme damit gelöst werden können, z. B. die Visualisierung von aus Experimenten resultierenden Gewichtungsvektoren und – weitaus bedeutender – die robuste Auswahl des richtigen Modells.

4 Struktur des Gewichtungsraums

Eine Mannigfaltigkeit ohne Rand
ist eine Mannigfaltigkeit mit Rand,
nur ohne Rand: $\partial M = \emptyset$.

— Prof. Dr. H. Scheerer, Vorlesung
„Analysis auf Mannigfaltigkeiten“

Es wird eine Symmetriegruppe S identifiziert, die auf den Gewichtungsraum wirkt ⟦Rüger und Ossen 1996a⟧. Ein Algorithmus zum Herausdividieren der Symmetriegruppe macht aus dem Gewichtungsraum ein Fundamentalgebiet, das mit einer geeigneten Metrik versehen werden kann ⟦Ossen und Rüger 1996b⟧. Mit Hilfe dieser Metrik können verschiedene Lernergebnisse im Fundamentalgebiet geclustert werden ⟦Rüger und Ossen 1996c⟧. Dieses Clustern kann z. B. dazu benutzt werden, den Prognosefehler bei Zeitreihen besser einzuschätzen ⟦Ossen und Rüger 1996d⟧. Es erscheint so, als ob diese Methode im allgemeinen die Beurteilung der Kombination von Datenmenge, Modellauswahl und Lernverfahren erlaubt ⟦Rüger und Ossen 1996e⟧. Es wird demonstriert, wie aufgrund von Cluster-Ergebnissen eine robuste Modellauswahl vorgenommen werden kann ⟦Ossen und Rüger 1996f⟧. Schließlich wird in diesem Kapitel das Clustered bootstrap eingeführt, das die Güteeigenschaften der Delta-Methode mit der Robustheit der Bootstrap-Methode kombiniert, mit dem Ziel einer guten Beurteilung der Variabilität der Netzfunktion.

4.1 Das Identifizierungsproblem

Verschiedene Gewichtungsvektoren können die gleiche Netzfunktion erzeugen. Sei ein Feedforward-Netz mit einer Eingabeschicht N_o, k Binnenschichten $N_1, \ldots, N_k$ und einer Ausgabeschicht N_{k+1} wie in Unterabschnitt (1.1.1) gegeben, wobei jeder Nichteingabe-Knoten a ein Biasgewicht w_{oa}

erhält. Dann ist Netzfunktion out_w beispielsweise invariant unter der Umbenennung von Knoten einer Binnenschicht N_i; dies liegt an der Kommutativität der Addition bei der Bestimmung der Nettoeingabe in Knoten der Schicht N_{i+1}.

Abb. 52 zeigt zum Beispiel die Projektion von 80 Gewichtungsvektoren, die durch 80faches Lernen ein und desselben Zusammenhangs aber mit verschiedenen Anfangsinitialisierungen zustande kamen. Es werden jedoch nur im wesentlichen vier verschiedene Netzfunktionen erzeugt (angezeigt durch vier verschiedenen Symbole), was der Lage der Gewichtungsvektoren keinesfalls anzusehen ist.

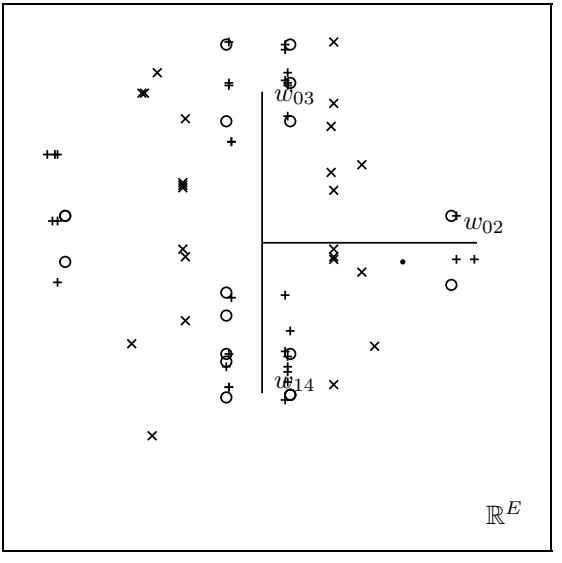


Abb. 52 Projektion von Gewichtungsvektoren

Symmetrie (4.1.1) Jede Abbildung $s\colon \mathbb{R}^E \to \mathbb{R}^E$, für die gilt, daß $\text{out}_w = \text{out}_{s(w)}$ für alle $w \in \mathbb{R}^E$, heiße *Symmetrie* des Gewichtungsraums. Sie läßt die Netzfunktion invariant. □

Beispiele (4.1.2) (i) Symmetrien durch Umbenennung von Binnenknoten innerhalb einer Schicht N_i sind also bestimmte Permutationen $\pi\colon \mathbb{R}^E \to \mathbb{R}^E$ der Komponenten des Gewichtungsvektors; Abb. 54a zeigt so ein Beispiel.

(ii) Typische Aktivierungsfunktionen f_b eines Binnenknoten b zeigen eine Punktsymmetrie $f_b(x) = e - f_b(-x)$ ihres Graphen bezüglich eines Zentrums $(0, e/2)$, wobei z. B. $e = 1$ für die Fermifunktion und $e = 0$ für tanh gilt. Dies induziert eine weitere Symmetrie des Gewichtungsraums: Sei $\tilde{w}(b, i)$ der Teilvektor von $w \in \mathbb{R}^E$, dessen Komponenten mit dem Knoten $b \in N_i$ zu tun haben (siehe Abb. 53), d. h. $\tilde{w}(b, i) = (w_{ob}, w_{a_1 b}, w_{a_2 b}, \ldots, w_{bc_1}, w_{bc_2}, \ldots)$. Hier sind a und c Aufzählungen der Knoten in N_{i-1} bzw. N_{i+1}. Wird das Vorzeichen aller Komponenten des Teilvektors $\tilde{w}(b, i)$ von w gewechselt, so kann dies korrigiert werden durch die Änderung $w_{oc_i} \mapsto w_{oc_i} + e \cdot w_{bc_i}$ aller Biasgewichte in der Schicht N_{i+1} (Abb. 54b). Diese Symmetrie (Vorzeichenwechsel und Kompensation) sei mit $t_b \colon \mathbb{R}^E \to \mathbb{R}^E$ notiert. t_b ist sein eigenes Inverses und wird daher auch *Spiegelsymmetrie* des Knotens b genannt; im Gegensatz dazu wird eine Symmetrie aus (i) *Permutationssymmetrie* genannt.

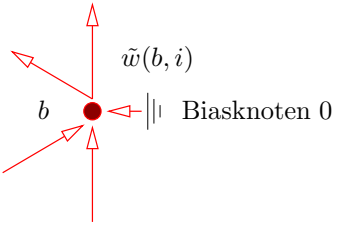


Abb. 53 Teilvektor des Gewichtungsvektors an einem Knoten $b \in L_i$

(iii) Sei wie in (ii) b aus einer Binnenschicht N_i und c eine Aufzählung der Knoten aus N_{i+1}. Dann ist folgende Abbildung

$$w \mapsto \left(aj \mapsto \begin{cases} 1 & \text{falls } j = b \text{ und } w_{bc_1} = w_{bc_2} = \ldots = 0 \\ w_{aj} & \text{sonst} \end{cases} \right).$$

eine Symmetrie. Es werden beim Gewichtungsvektor w alle beim Knoten b einlaufenden Gewichte auf Eins gesetzt, wenn alle auslaufende Gewichte verschwinden und ansonsten alle Komponenten von w unverändert gelassen werden. Sind nämlich alle auslaufenden Gewichte eines Knotens b null, dann sind zur Bestimmung der Netzfunktion die einlaufenden Gewichte

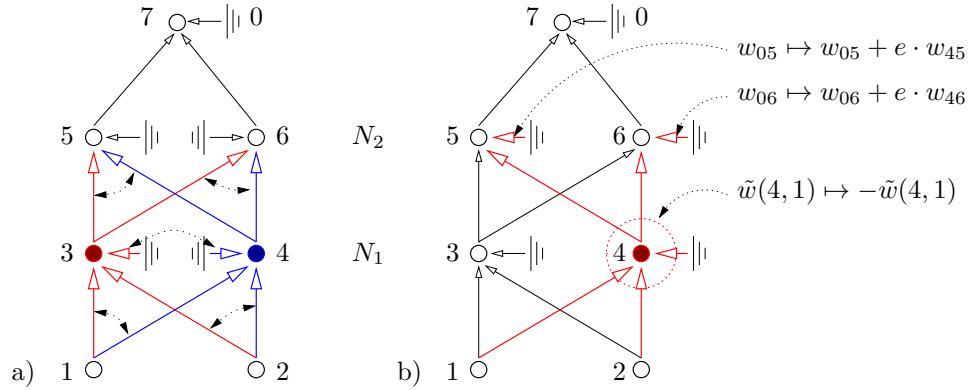


Abb. 54 Symmetrien von $w \mapsto \mathrm{out}_w$ im Gewichtungsraum induziert durch a) eine gewisse Permutation $\pi\colon \mathbb{R}^E \to \mathbb{R}^E$ — infolge Umbenennung der Knoten 3 und 4 — und b) durch die Symmetrie $f(x) = e - f(-x)$ der Aktivierungsfunktion

irrelevant; sie können daher auf einen beliebigen Wert, z. B. Eins gesetzt werden. □

Relevanz der Beispiele (4.1.3) Sowohl die Permutationssymmetrien als auch die Spiegelsymmetrien sind bijektive lineare Operatoren auf $\mathbb{R}^E$, ja sogar analytisch. In ⟦Chen et al. 1993⟧ wird gezeigt, daß dies schon alle analytischen Symmetrien sind.

Im Gegensatz dazu ist die Symmetrie (4.1.2;iii) nicht stetig. Alle Gewichtungsvektoren bis auf eine Menge vom Lebesgue-Maß Null werden auf sich abgebildet. Die für die Unstetigkeit verantwortliche Ausnahmemenge wird nach ⟦Sussmann 1992⟧ genau dadurch charakterisiert, daß Vektoren in ihr Netze erzeugen, die reduzibel sind, d. h. mit weniger Knoten die gleiche Netzfunktion darstellen können. Die Menge der Gewichtungsvektoren, die reduzible Netze erzeugen, ist so durch einige einfache lineare Gleichungen gegeben ⟦Sussmann 1992⟧, daß ihr Lebesgue-Maß verschwindet. Daher reicht es, sich auf die Permutations- und Spiegelsymmetrien zu konzentrieren. □

4.2 Die relevante Symmetriegruppe S

Im folgenden werden neuronale Feedforward-Netze mit $k > 0$ Binnenschichten betrachtet; sämtliche Binnenknoten seien mit einer nichtlinearen und gemäß $f(x) = e - f(-x)$ punktsymmetrischen Aktivierungsfunktion ausgestattet. Die Permutations- und Spiegelsymmetrien sind Elemente der Gruppe $\mathrm{GL}(E, \mathbb{R})$ von bijektiven und linearen Funktionen $g: \mathbb{R}^E \to \mathbb{R}^E$, wobei die Gruppenmultiplikation die Hintereinanderausführung der Funktionen bedeute. Daher erzeugen die Symmetrien eine Untergruppe S von $\mathrm{GL}(E, \mathbb{R})$. Die Gruppentheorie erscheint als Mittel der Wahl, um diese Symmetrien zu beschreiben.

Permutationssymmetrie-Gruppe Π_i (4.2.1) Sei $\Sigma(M)$ die Permutationsgruppe einer Menge M. Jede Permutation kann geschrieben werden als Hintereinanderausführung von *Transposition*en, d. s. Vertauschungen von Nachbarelementen. Die Transposition $\tau(a, i)$ eines Binnenknoten $a \in N_i$ mit seinem rechten Nachbarknoten induziert eine bestimmte Permutation $\pi(a, i)$ der Gewichte, so daß die Netzfunktion invariant bleibt (wie in Abb. 54a).

Die Abbildung $\tau(a, i) \mapsto \pi(a, i)$ erzeugt mit variierendem a und festem i einen natürlichen Monomorphismus, d. i. ein injektiver Homomorphismus. Sei Π_i das Bild dieses Monomorphismus. Π_i kann identifiziert werden mit der durch die $|N_i| - 1$ Elemente $\pi(a_1, i)$, $\pi(a_2, i)$, ... *erzeugten Gruppe* von $\mathrm{GL}(E, \mathbb{R})$, d. i. die kleinste Untergruppe von $\mathrm{GL}(E, \mathbb{R})$, die alle $\pi(a_1, i)$, $\pi(a_2, i)$, ... enthält. □

Spiegelsymmetrie-Gruppe T_a (4.2.2) Sei a ein Binnenknoten. Da die in (4.1.2;ii) definierte Abbildung t_a ihr eigenes Inverses ist, erzeugt sie die zyklische Gruppe $T_a := \{1, t_a\}$. □

Symmetriegruppe S (4.2.3) Die Symmetriegruppe S sei die durch Π_1, Π_2, ..., Π_k und alle T_a erzeugte Untergruppe von $\mathrm{GL}(E, \mathbb{R})$. □

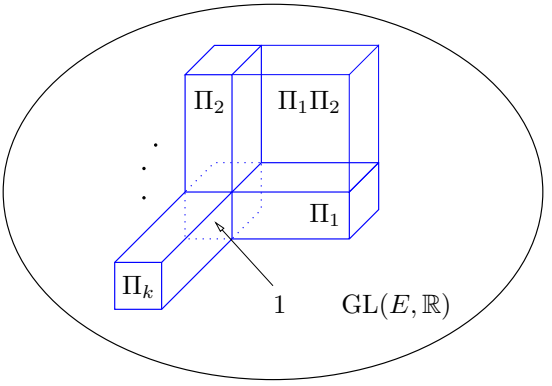


Abb. 55 Struktur der Permutationsgruppen als Untergruppen von $\mathrm{GL}(E,\mathbb{R})$

Theorem: Struktur von S (4.2.4) *Jedes Element von S hat eine eindeutige Darstellung $s = \pi_k \dots \pi_1 \prod_b \tau_b$ mit $\pi_i \in \Pi_i$, $\tau_b \in T_b$ und $b \in N_1 \cup \dots N_k$.* □

Beweis (4.2.5) Zunächst überzeuge man sich davon, daß Π_i mit Π_j elementeweise kommutiert, falls $i \neq j$. Des weiteren kann kein Element von Π_j (verschieden von der Identität) von irgendeiner Kombination von Elementen aus $\Pi_1 \dots \Pi_{j-1}\Pi_j \dots \Pi_k$ erzeugt werden. Diese Situation, in Abb. 55 visualisiert, erlaubt die Anwendung eines Theorems aus der Gruppentheorie ⟦Macdonald 1975, Theorem 4.39⟧: es besagt, daß die Gruppe Π, die von Π_1, …, Π_k erzeugt wird, isomorph ist zum direkten Produkt der Gruppen Π_1, …, Π_k. Insbesondere hat diese Gruppe $|N_1|!|N_2|! \dots |N_k|!$ Elemente.

Im nächsten Schritt überzeuge man sich, daß t_b mit t_a kommutiert für alle Binnenknoten a und b; zudem kann offensichtlich kein t_b durch irgendeine Kombination von t_{a_1}, t_{a_2}, … erzeugt werden, falls nur alle a_i verschieden von b sind. Mit dem gleichen Argument wie vorher ist die Gruppe T, die von allen t_b erzeugt wird, isomorph zum direkten Produkt aller Gruppen T_b. Insbesondere hat diese Gruppe $2^{|N_1 \cup \dots \cup N_k|}$ Elemente.

Obwohl nun T und Π nicht elementeweise kommutieren, ist doch $\pi^{-1}t\pi$ für alle $\pi \in \Pi$ und $t \in T$ wieder in T. Daher kommutieren Π und T als Mengen: $\Pi T = T\Pi$. Nach ⟦Macdonald 1975, Korollar 6.09⟧ gilt dann $S = \Pi T$. Zusammen mit obiger Beobachtung, daß T das direkte Produkt der T_b und Π das direkte Produkt der Π_i ist, folgt die Behauptung. ■

4.3 Ein Fundamentalgebiet W

Wirkung und Fundamentalgebiet (4.3.1) Die *Wirkung* einer Gruppe G auf eine Menge M ist eine Abbildung $G \times M \to M$, $(g, m) \mapsto gm$, die $(g_1 g_2)m = g_1(g_2 m)$ und $1m = m$ für alle $g_1, g_2 \in G$ und $m \in M$ erfüllt. Die Symmetriegruppe S wirkt auf $\mathbb{R}^E$ in natürlicher Weise durch $(s, w) \mapsto s(w)$. Verschiedene *Orbits* $S(w) = \{x \in \mathbb{R}^E \mid x = s(w) \text{ für ein } s \in S\}$ partitionieren $\mathbb{R}^E$. Jede offene und konvexe Menge $W \subset \mathbb{R}^E$, in der von jedem Orbit höchstens ein *Repräsentant* (Element) enthalten ist, so daß $S(W)$ dicht in $\mathbb{R}^E$ ist, heiße *Fundamentalgebiet.* □

Theorem: Fundamentalgebiet W (4.3.2) *Sei a^i eine Aufzählung der Knoten der Binnenschicht N_i. Dann ist*

$$W := \{w \in \mathbb{R}^E \mid 0 < w_{oa_1^i} < \ldots < w_{oa^i_{|N_i|}} \text{ für alle } 1 \leq i \leq k\}$$

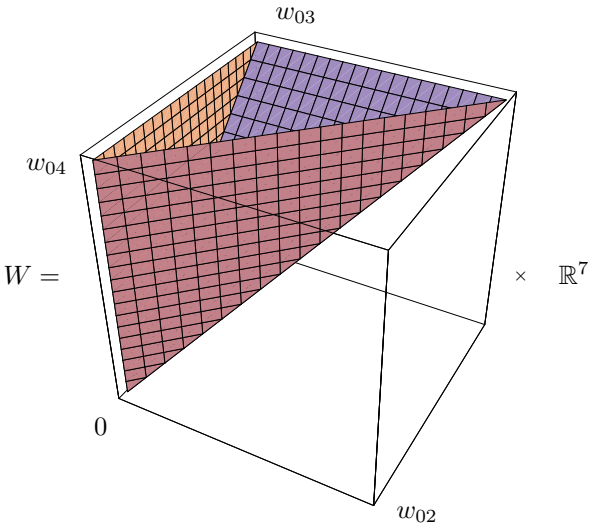
ein Fundamentalgebiet. □

Beweis (4.3.3) Daß W offen und konvex ist, bedarf kaum eines Beweises. Wie in (4.2.5) gesehen, zerfällt S in ΠT. Da jedes nichttriviale $t \in T$ mindestens ein Vorzeichen eines Biasgewichts ändert (was keinesfalls durch eine Permutation $\pi \in \Pi$ rückgängig gemacht werden kann), gilt $t(w) \notin W$, falls $w \in W$ und $t \in T \setminus \{1\}$. Andererseits gilt, daß jede nichttriviale Permutation $\pi \in \Pi$ die Reihenfolge der Biasgewichte zerstört. Es gibt also kein $s \in S \setminus \{1\}$ und kein $w \in W$, so daß $s(w) \in W$. Hat also ein Orbit von $w \in \mathbb{R}^E$ einen Repräsentanten in W, dann ist das schon der einzige.

Wird nun T auf W angewendet, so erlaubt das jedes Vorzeichen von Biasgewichten. Die dann noch bestehende Restriktion der Anordnung der absoluten Biasgewichte wird durch die weitere Anwendung von Π entfernt. $S(W) = \Pi T(W)$ ist dann nur noch durch die Bedingung charakterisiert, daß die Biasgewichte von Binnenknoten ungleich null zu sein haben und daß zwei Biasgewichte einer Binnenschicht nicht gleichen Betrag haben dürfen. Wird nun der Abschluß dieser Menge genommen, so fällt auch diese Bedingung weg und es verbleibt $\mathbb{R}^E$. ■

Bemerkungen (4.3.4) (i) Es reicht also, in W anstelle des viel größeren Raums $\mathbb{R}^E$ zu arbeiten. Die Idee eines Fundamentalgebiets ist es, eine bequeme, in bezug auf S nichtredundante Teilmenge von $\mathbb{R}^E$ zu definieren.

(ii) W hat die Form eines Kegels mit Spitze bei 0, d. h. $cw \in W$ für alle $w \in W$ und $c > 0$. Abb. 56 zeigt ein Beispiel. □



$$W = \left\{ w \in \mathbb{R}^{10} \,|\, 0 < w_{02} < w_{03} < w_{04} \right\}$$

Abb. 56 Visualisierung des Fundamentalgebiets W eines 1-3-1-Netzes

Algorithmus für Repräsentanten in W (4.3.5) Sei $\hat{w} \in \mathbb{R}^E$ ein Gewichtungsvektor, wie er beispielsweise durch einen Lernalgorithmus oder eine Parameterschätzung zustande gekommen ist.

Beginnend mit der Schicht N_1 wende die Symmetrie t_a an, falls ein Biasgewicht negativ ist. Damit wird erreicht, daß alle Biasgewichte nichtnegativ sind. Wende danach Schicht für Schicht Permutationssymmetrien an, so daß die Biasgewichte in jeder Schicht in einer definitiven Reihenfolge, z. B. nichtabfallend, sind. Dies ist im wesentlichen ein einfaches Sortierproblem. Durch Verwendung eines stabilen[♮] Sortieralgorithmus wird ein eindeutiger

[♮] d. h. daß bei Gleichheit der Biasgewichte zweier Knoten, deren Reihenfolge durch das Sortieren unverändert bleibt

Repräsentantenvektor $r(\hat{w})$ erzeugt; er liegt im Abschluß $\overline{W}$ vom Fundamentalgebiet W. □

4.4 Die metrische Struktur von W

Der Gewichtungsraum als Mannigfaltigkeit (4.4.1) Oft entsteht eine Mannigfaltigkeit, wenn Punkte miteinander identifiziert werden. Sei zum Beispiel $(x_1, x_2) \mapsto -(x_1, x_2)$ eine Symmetrie auf $\mathbb{R}^2$. Dann kann $\mathbb{R}_o^+ \times \mathbb{R}$ als Abschluß eines Fundamentalgebiets betrachtet werden. Durch Verkleben äquivalenter Randpunkte entsteht aus dem flachen $\mathbb{R}^2$ ein geometrisch völlig neues Gebilde, in diesem Fall ein Kegel (siehe Abb. 57). Damit diesem Gebilde überall eine sinnvolle differenzierbare Struktur aufgeprägt werden kann, muß noch der Ursprung (die Spitze des Kegels) als singulärer Punkt entfernt werden. So entsteht durch die einfache Symmetrie eine gekrümmte Mannigfaltigkeit mit einem singulären Punkt.

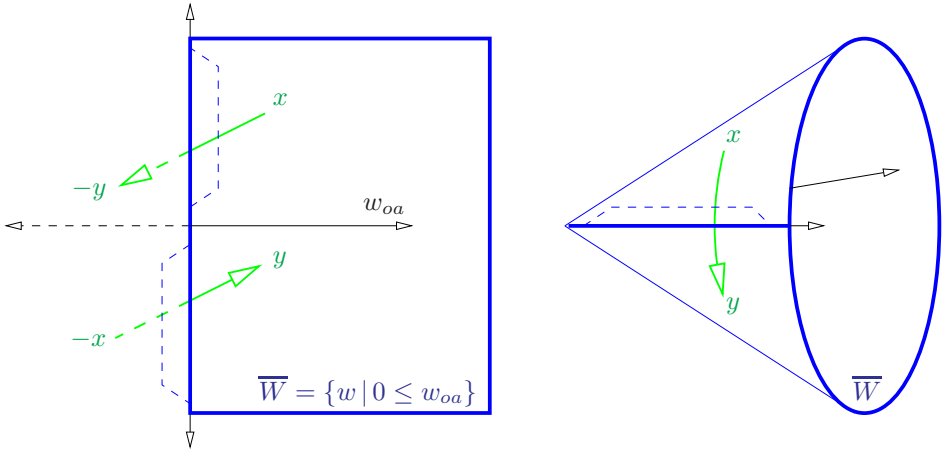


Abb. 57 $\overline{W} \setminus \{0\}$ als Mannigfaltigkeit

Diese Idee läßt sich ebenso auf die relevanten Symmetrien des Gewichtungsraums anwenden; obiges Beispiel entspricht ja gerade der Spiegelsymmetrie eines Binnenknotens. Man erhält dann eine gekrümmte Mannigfaltigkeit $M := (\mathbb{R}^E \setminus X)/S$ bestehend aus verschiedenen Orbits. X bezeichnet hier

die Menge der Gewichtungsvektoren mit einem verschwindenden Teilvektor $\tilde{w}(a,i)$ oder mit zwei übereinstimmenden Teilvektoren $\tilde{w}(a,i) = \tilde{w}(b,i)$ innerhalb einer Binnenschicht. Alle Orbits von Vektoren in $\mathbb{R}^E \setminus X$ haben dann die gleiche Mächtigkeit $|S|$. Schließlich bedeute die Division durch die Symmetriegruppe die Identifizierung äquivalenter Vektoren. Die singulären Punkte dieser so erhaltenen Mannigfaltigkeit M kommen durch die Herausnahme der Punkte in X zustande. Im Prinzip veranschaulicht Abb. 57 eine Untermannigfaltigkeit von M.

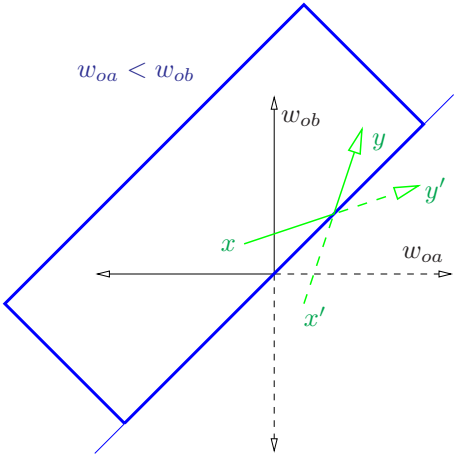


Abb. 58 Reflexion am Rand der Mannigfaltigkeit

Nun taucht das Problem auf, wie Lernalgorithmen oder Statistik auf der Mannigfaltigkeit M definiert werden können, die ja völlig andere geometrische Eigenschaften hat als der flache Raum $\mathbb{R}^E$. Eine Möglichkeit wäre, Lernregeln als Differentialgleichungen zu formulieren, z. B. wie in (2.3.1) als $\partial w/\partial t = -\nabla E(w)$. Diese Differentialgleichungen könnten dann auf M gelöst werden. Eine andere Möglichkeit ist es, nach jedem Lernschritt (nach jeder Änderung des Gewichtungsvektors von x zu y) die richtige Symmetrieoperation anzuwenden, wenn der Gewichtungsvektor bei seiner Bewegung den Rand der Mannigfaltigkeit trifft. Dabei kommt es zu Reflexionen, wie in Abb. 58 zu sehen ist.

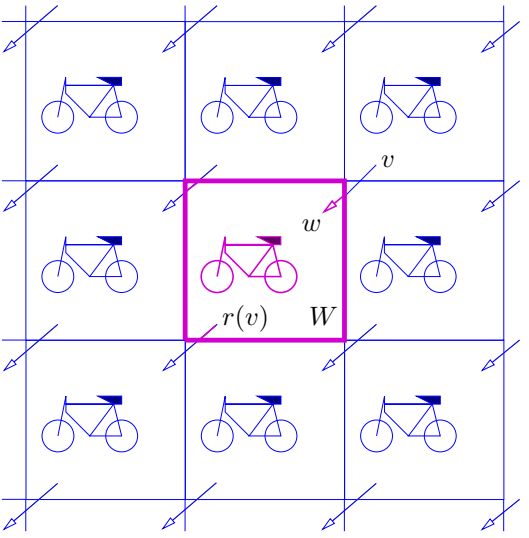


Abb. 59 Die euklidische Metrik im Fundamentalgebiet gibt die Abstände nicht richtig wieder: $\mathbb{R}^2$ ist zusammen mit den Symmetrien $(x, y) \mapsto (x+1, y)$ und $(x, y) \mapsto (x, y+1)$ dargestellt; das Einheitsquadrat ist dann ein natürliches Fundamentalgebiet und die Fahrräder symbolisieren äquivalente Orte; Spitze w und Fuß v des eingezeichneten Pfeils sind sehr nahe beieinander, jedoch erscheinen ihre Repräsentanten unter der euklidischen Metrik im Fundamentalgebiet weit entfernt

Eine weitere, im folgende bevorzugte Möglichkeit ist es, im flachen Raum zu lernen und das Lernergebnis in den nichtredundanten Raum $\overline{W}$ abzubilden; das erspart die Beschäftigung mit Geodätischen, Paralleltransport und Krümmung. □

Notwendigkeit einer eigenen Metrik für W (4.4.2) Für viele Anwendungen ist die differenzierbare Struktur einer Mannigfaltigkeit nicht nötig. Um die *Ortsinformation* eines bestimmten Gewichtungsvektors im Fundamentalgebiet zu nutzen, sollte aber zumindest eine geeignete Metrik definiert werden. Die reguläre euklidische Metrik erweist sich als unzureichend: zwei Punkte $v \notin W$ und $w \in W$ können in $\mathbb{R}^E$ sehr nahe sein, wobei ihre Repräsentanten $r(v)$ und $w = r(w)$ weit entfernt sind (siehe Abb. 59). Eine Idee ist es, als Abstand zweier Punkte v und w den minimalen Abstand ihrer Orbits $S(v)$ und $S(w)$ zu definieren. □

Dies erfordert jedoch eine weitere Struktureigenschaft der zugrundeliegenden Metrik im Gewichtungsraum:

Kompatibilität von Symmetrie und Metrik (4.4.3) Eine Metrik d_E von $\mathbb{R}^E$ heiße *kompatibel mit der Symmetriegruppe* S, wenn für alle $s \in S$ und $v, w \in \mathbb{R}^E$ gilt $d_E(v, w) = d_E(sv, sw)$. □

Theorem: Metrik in W (4.4.4) *Wenn eine Metrik d_E von $\mathbb{R}^E$ kompatibel mit S ist, dann ist durch*

$$d(v, w) := \min_{s \in S} d_E(sv, w)$$

eine Metrik auf W gegeben. □

Beweis (4.4.5) Da d_E eine Metrik ist, verschwindet $d(v, w)$ genau dann, wenn $w \in S(v)$. Weil aber v und w in W – einem Fundamentalgebiet – liegen, müssen dann aber v und w zusammenfallen. Jede Symmetrie $s \in S$ ist nach Voraussetzung eine Isometrie auf $\mathbb{R}^E$; daher ergibt sich $d(v, w) = \min d_E(sv, w) = \min d_E(v, s^{-1}w) = d(w, v)$. Des weiteren muß es $s_1, s_2 \in S$ geben mit $d(v, y) + d(y, w) = d_E(s_1 v, y) + d_E(y, s_2^{-1} w)$, was die Anwendung der Dreiecksungleichung für d_E erlaubt; dies resultiert in der Verkleinerung zu $d_E(s_1 v, s_2^{-1} w) = d_E(s_2 s_1 v, w)$, was wiederum aufgrund der Definition von d durch $d(v, w)$ minorisiert wird. ■

Mit S kompatible Metriken (4.4.6) (i) Wenn die Aktivierungsfunktionen der Binnenschichten punktsymmetrisch zum Ursprung sind, wie z. B. bei der tanh-Funktion, dann sind die euklidische Metrik, die Manhattan-Metrik, die Maximumsmetrik usw. alle kompatibel mit S: alle Punkte des Orbits von w liegen auf einer Kugel um den Ursprung mit Radius $|w|$.

(ii) Wenn im Gegensatz dazu die Fermifunktion als Aktivierungsfunktion benutzt wird, also wenn gilt $f(x) = 1 - f(-x)$, dann muß ein Vorzeichenwechsel eines Teilvektors $\tilde{w}(a, i)$ durch Biaskorrekturen in der nächsten Schicht N_{i+1} ausgeglichen werden. Dies verändert auch die euklidische Norm des äquivalenten Vektors. Es gibt in diesem Fall aber eine Norm

$|\bullet|_E$, die eine mit S kompatible Metrik induziert. Ist nur eine Binnenschicht beteiligt, so kann z. B.

$$|w|_E = \max_{d\in N_2}\Big\{|w|_\infty, \big|w_{od} + \sum_{c\in N_1}\max(0,w_{cd})\big|, \big|w_{od} + \sum_{c\in N_1}\min(0,w_{cd})\big|\Big\}$$

gewählt werden, wobei $|\bullet|_\infty$ die Maximumsnorm in $\mathbb{R}^E$ bezeichnet. Die Einheitskugel in dieser speziellen Metrik ist so verzerrt, daß alle Punkte eines Orbits auf der gleichen Kugeloberfläche liegen. Diese Norm könnte eine natürliche Wahl für eine Metrik $d_E(v,w) = |v-w|_E$ sein, wenn die Fermifunktion benutzt wird. Die Verzerrung der Einheitskugel reflektiert die Bevorzugung positiver Ausgabewerte der Fermifunktion. Es fällt nicht schwer, die Metrik auf mehr als eine Binnenschicht oder auf den Symmetrietyp $f(x) = e - f(-x)$ zu erweitern. □

Jedoch scheint es auch keinen Grund dafür zu geben, auf der Wahl der Fermifunktion zu bestehen. Deswegen wird im folgenden die Verwendung von *zum Ursprung* punktsymmetrischen Aktivierungsfunktionen – wie tanh – angenommen.

Intraktabilität der Berechnung (4.4.7) *Die Berechnung von d ist ein NP-hartes Problem.* □

Beweis (4.4.8) Es genügt eine spezielle Konfiguration anzugeben, die NP-hart ist. Sei z. B. $d_E(v,w) = |v-w|_1$ die Manhattan-Metrik in $\mathbb{R}^E$ und tanh die Aktivierungsfunktion aller Binnenknoten. Das Netz habe nur eine Binnenschicht. Bei nur einer Binnenschicht gibt es keine Verbindungen zwischen Binnenknoten; daher gehört jede Gewichtungskomponente w_{ab} entweder zu $\tilde{w}(a,1)$ falls $a \in N_1$, zu $\tilde{w}(b,1)$ falls $b \in N_1$ oder er ist ein Biasgewicht der Schicht N_2. Letztere werden in dem Teilvektor $\tilde{w}_o$ zusammengefaßt. Der Teilvektor $\tilde{w}(a,1)$ wird im folgenden als $\tilde{w}_a$ abgekürzt, $a \in N_1$. Aus der Definitionsgleichung $d(v,w) = \min d_E(sv,w)$ ergibt sich konkret

$$\begin{aligned} d(v,w) &= \min_{\pi\in\Pi_1,t\in T}|t\pi v - w|_1 = \min_{\pi\in\Pi_1,t\in T}|\pi v - tw|_1 \\ &= |\tilde{v}_o - \tilde{w}_o|_1 + \min_{\pi\in\Pi_1,t\in T}\sum_{a\in N_1}|\widetilde{\pi v}_a - \widetilde{tw}_a|_1 \end{aligned}$$

$$= |\tilde{v}_o - \tilde{w}_o|_1 + \min_{\pi\in\Pi_1} \sum_{a\in N_1} \min_{t\in T_a} |\widetilde{\pi v}_a - \widetilde{tw}_a|_1$$
$$= |\tilde{v}_o - \tilde{w}_o|_1 + \min_{\pi\in\Sigma(N_1)} \sum_{a\in N_1} \min(|\tilde{v}_{\pi a} - \tilde{w}_a|_1, |\tilde{v}_{\pi a} + \tilde{w}_a|_1).$$

In der ersten Zeile wird sowohl die eindeutige Zerlegbarkeit (4.2.4) von Elementen der Symmetriegruppe S benutzt als auch die Isometrieeigenschaft der Spiegelsymmetrien in Zusammenhang mit der Tanh-Aktivierungsfunktion. Danach werden die Vektoren v und w in Teilvektoren zerlegt; dabei wird ausgenutzt, daß $\tilde{v}_o$ und $\tilde{w}_o$ aus dem Minimierungsprozeß herausgezogen werden können, weil sie definitionsgemäß invariant unter Symmetrien sind. Zudem wirken Spiegelsymmetrien lokal auf die Teilgewichtungsvektoren; bezüglich dieser Symmetrien können also die Summanden einzeln minimiert werden. Schließlich kann die Permutationssymmetrie im Gewichtungsraum mittels ihres definierenden natürlichen Monomorphismus direkt auf eine Permutation der Binnenknoten übertragen werden. Der zweite Summand

$$\min_{\pi\in\Sigma(N_1)} \sum_{a\in N_1} K(a, \pi(a)) \tag{K}$$

der letzten Zeile mit $K(a,b) := \min(|\tilde{w}_a - \tilde{v}_b|_1, |\tilde{w}_a + \tilde{v}_b|_1)$ ist von der Struktur des allgemeinen Problems des Handlungsreisenden. Dies ist ein NP-hartes Problem im Sinn von ⟦Garey und Johnson 1979⟧. ∎

Bemerkungen (4.4.9) (i) Obiger Beweis kann leicht auf eine viel größere Klasse von Metriken erweitert werden: sei $d_E(v,w) = |v-w|_p$ durch die p-Norm ($p \geq 1$) induziert. Dann ist $d(v,w)^p - |\tilde{v}_o - \tilde{w}_o|_p^p$ von der gleichen Struktur wie (4.4.8;K) mit $K(a,b) := \min(|\tilde{w}_a - \tilde{v}_b|_p^p, |\tilde{w}_a + \tilde{v}_b|_p^p)$.

(ii) Das allgemeine Problem des Handlungsreisenden ist typischerweise auf einer Menge M von n Städten definiert mit einer (nicht notwendigerweise symmetrischen) Kostenmatrix $D\colon M\times M \to \mathbb{R}$. Das Element $D(a,b)$ enthält die Reisekosten zwischen den Städten a und b. Sei $\rho\colon M \to M$, $m_i \mapsto m_{(i+1) \bmod n}$ die Permutation, die jeder Stadt die nächste bezüglich einer fest gewählten Aufzählung $m_o, m_1, \ldots, m_{n-1}$ zuordnet. Es gilt nun

die Kosten einer Rundreise durch alle Städte zu minimieren, also eine Permutation $\pi \in \Sigma(M)$ zu finden, die

$$\sum_{a\in M} D(\pi a, \pi\rho a) = \sum_{a\in M} D(a, \pi\rho\pi^{-1}a)$$

optimiert. Nun durchlaufen die von π erzeugten zu ρ *konjugierten Elemente* $\pi\rho\pi^{-1}$ nicht die gesamte Permutationsgruppe $\Sigma(M)$, sondern nur eine bestimmte Teilmenge: alle von $\pi\rho^i$ erzeugten zu ρ konjugierten Elementen stimmen per Definition überein. ρ ist eine Art Rotationsoperator; für ihn gilt offensichtlich $\rho^n = 1$. Unter den n verschiedenen Permutationen $\pi, \pi\rho, \ldots, \pi\rho^{n-1}$ gibt es genau eine, die die Stadt m_o invariant läßt. Sei $\Sigma^{m_o}(M)$ diejenige Untergruppe von $\Sigma(M)$, deren Elemente die Stadt m_o fest läßt (sie ist isomorph zu $\Sigma(M \setminus \{m_o\})$. Dann reicht es also schon,

$$\operatorname*{argmin}_{\pi\in\Sigma^{m_o}(M)} \sum_{a\in M} D(a, \pi\rho\pi^{-1}a) \qquad (D)$$

zu bestimmen. Dies entspricht der Anschauung, daß nur $(n-1)!$ verschiedene Anordnungen von n auf einem Kreis angeordneten Städten existieren.

Obwohl (D) für $n = |N_1| + 1$ Städte insofern schon mit (4.4.8;K) übereinstimmt, als daß ein Minimum von $(n-1)!$ Termen gebildet wird, ist die $n \times n$-Matrix D, deren Diagonale nie benutzt wird, in seiner Bedeutung völlig verschieden von der $(n-1)\times(n-1)$-Matrix K.

Am einfachsten läßt sich (4.4.8;K) direkt als Problem des Handlungsreisenden interpretieren mit $n = 2|N_1|$ Städten $\hat{w}_a$, $\hat{v}_a$, $a \in N_1$, wobei $D(\hat{w}_a, \hat{w}_b) = D(\hat{v}_a, \hat{v}_b) = \infty$, $D(\hat{w}_a, \hat{v}_b) = K(a,b)$ und $D(\hat{v}_a, \hat{w}_b) = 0$, also kurz mit der Kostenmatrix ⟦Ossen 1996⟧

$$D = \begin{pmatrix} \infty & K \\ 0 & \infty \end{pmatrix}.$$

Dabei sind die $\hat{w}_a$ Talstationen und die $\hat{v}_b$ Berggipfel in einem besonderen Skigebiet: weder zwischen denen Talstationen gibt es (Ski-) Wege noch zwischen den Berggipfeln. Von jeder Talstation $\hat{w}_a$ aus gibt es Wege zu jedem Berggipfel $\hat{v}_b$ mit Liftkosten $K(a,b)$. Die Talfahrt ist von jedem Berggipfel aus zu beliebigen Talstationen möglich und umsonst. Die Kosten einer Rundreise sind dann schon durch die Angabe, von welcher Talstation aus zu welchem Gipfel gefahren wird, gegeben. □

Effiziente Approximation (4.4.10) Daher ist es möglich, die reichhaltige Literatur zur effizienten Approximation des Problems des Handlungsreisenden ⟦z. B. Lawler et al. 1985 und Referenzen darin⟧ zur Implementierung einer Approximation d' von d auszunutzen: Sind v und w zufällig, dann ergibt in meiner Implementierung $d'(v,w)$ den richtigen Wert $d(v,w)$ in 99,8% der Fälle (empirisch bei weniger als acht Binnenknoten geprüft); in den restlichen 0,2% überschritt der Wert von $d'(v,w)/d(v,w)$ nie den Faktor 1,2. Des weiteren verschwindet $d'(v,w)$ immer, wenn v und w äquivalent sind (irgend eine Anzahl von Knoten). Als Nebenprodukt wird ein Gewichtungsvektor $\mathrm{nearest}(v,w) \in S(v)$ bestimmt, für den $d'(v,w) = d_E(\mathrm{nearest}(v,w),w)$ gilt; mithin ist er mutmaßlich dasjenige Element aus dem Orbit von v, das w am nächsten kommt. □

Weitere kanonische Metriken (4.4.11) (i) Andere Metriken werden z. B. durch

$$(v,w) \mapsto d_1(v,w) := \sup_{x\in X} |\,\mathrm{out}_w(x) - \mathrm{out}_v(x)|$$

oder etwa durch

$$(v,w) \mapsto d_2(v,w) := \sqrt{\int_X (\mathrm{out}_w(x) - \mathrm{out}_v(x))^2 dx}$$

gegeben. Dabei ist $X \subset \mathbb{R}^{N_o}$ eine typischerweise kompakte Menge im Eingaberaum. Es müssen nur noch Punkte mit Abstand Null miteinander identifiziert werden; das schließt – wie vorher – ein, Punkte im Orbit unter S zu identifizieren. Jedoch mag die Menge X Ursache für weitere Symmetrien sein. Der resultierende metrische Raum heiße $(\tilde{W}, d_i)$; dann sind obige Metriken – wie auch d – *kanonisch* in dem Sinn, daß die zugehörige, die Symmetrien herausdividierende Abbildung $r\colon (\mathbb{R}^E, d_E) \to (\tilde{W}, d_i)$ stetig ist. Ziel des gesamten Abschnitts 4.4 war es, eine praktikable kanonische Metrik zu definieren.

(ii) Die Metrik d_1 hat beispielsweise eine sehr klare und befriedigende Interpretation: die von v erzeugte Netzfunktion eingeschränkt auf X liegt in einem Epsilonschlauch der Größe $d_1(v,w)$ um die von w erzeugte Netzfunktion. d_1 wie auch d_2 sind Abstände im Funktionenraum der auf X eingeschränkten stetigen Funktionen nach $\mathbb{R}^O$. Jedoch stellt i. allg. die

tatsächliche Berechnung von d_1 oder d_2 ein aufwendiges und weit unhandhabbareres Problem als das der Berechnung von d.

(iii) Wegen der Stetigkeit von $w \mapsto \mathrm{out}_w$ kommt sehr kleinen Werten von $d(v, w)$ auch noch die Bedeutung zu, daß die erzeugten Netzfunktionen out_v und out_w sehr ähnlich sind. Es kann aber auch schon sein, daß sehr ähnliche Netzfunktionen eingeschränkt auf den Einheitswürfel einen großen Abstand aufweisen: Betrachte ein Netz mit einem Eingabe-, einem Binnen- und einem Ausgabeknoten mit Namen 1, 2, und 3. Das Gewicht w_{12} vermittelt also zwischen Eingabe- und Binnenknoten. Seien v und w zwei bis auf die 12-Komponente gleiche Gewichtungsvektoren mit $v_{12} = 100$ und $w_{12} = 90$. Der Abstand $d(v, w)$ ist 10, obwohl sich (wegen der schnellen Sättigung der Tanh-Aktivierungsfunktion) die Funktionen out_v und out_w nahezu gleichen.

Es mag daher sinnvoll erscheinen, eine heuristische kanonische Metrik h zu definieren, für die gilt $h(v, w) := d(\tilde{v}, \tilde{w})$ mit

$$\tilde{w}_{ab} = \begin{cases} w_{ab}/|L_a| & \text{falls } b \in N_{k+1} \\ \tanh(w_{ab}/|L_a|) & \text{falls } b \in N_2 \cup \ldots \cup N_k \\ \tanh(w_{ab}/|L_a|/\max|p|) & \text{falls } b \in N_1. \end{cases}$$

$\max|p|$ notiert hier die typische Größe eines Eingabevektors, z. B. Eins, wenn die Eingabevektoren aus dem Einheitswürfel kommen. Die Division durch den Fan-in $|L_a|$ wurde auch schon in Abschnitt (1.4.6) motiviert. Durch die Transformation $w \mapsto \tilde{w}$ werden die Komponenten des Gewichtungsvektors zunächst gemäß ihrer Bedeutung „zurechtgestutzt“. Dem Abstand $h(v, w)$ kommt dann eine erheblich bessere Interpretation zu als $d(v, w)$. □

4.5 Clustern im Gewichtungsraum

Praktischer Nutzen eines Fundamentalgebiets (4.5.1) Der Abschluß $\overline{W}$ des Fundamentalgebiets ist eine Art *effektiver Gewichtungsraum*. Ein typisches Experiment wäre etwa, mit einem Lernalgorithmus oder einer statistischen Schätzung einen guten Gewichtungsvektor mit bestimmten Eigenschaften wie gute Approximation einer Datenmenge zu suchen.

Hier kann durch Einschränkung auf $\overline{W}$ zwar der Suchraum um den Faktor $1/|S|$ verkleinert werden. Jedoch reduziert sich in gleicher Weise die Zahl der Lösungen. Daher erscheint es unplausibel, daß die alleinige Beschränkung auf das Fundamentalgebiet Fortschritte erzielt.[♮]

Die Situation ändert sich aber mit mehr als einem Lauf des Experiments, z. B. mit verschieden initialisierten anfänglichen Gewichtungsvektoren oder künstlich hinzugefügtem Rauschen. Hier können dann die resultierenden Gewichtungsvektoren w^1, w^2, ..., w^m bezüglich einer kanonischen Metrik im effektiven Gewichtungsraum gruppiert (geclustert) werden. Dies gibt Aufschluß über die typischen Antworten eines Experiments. □

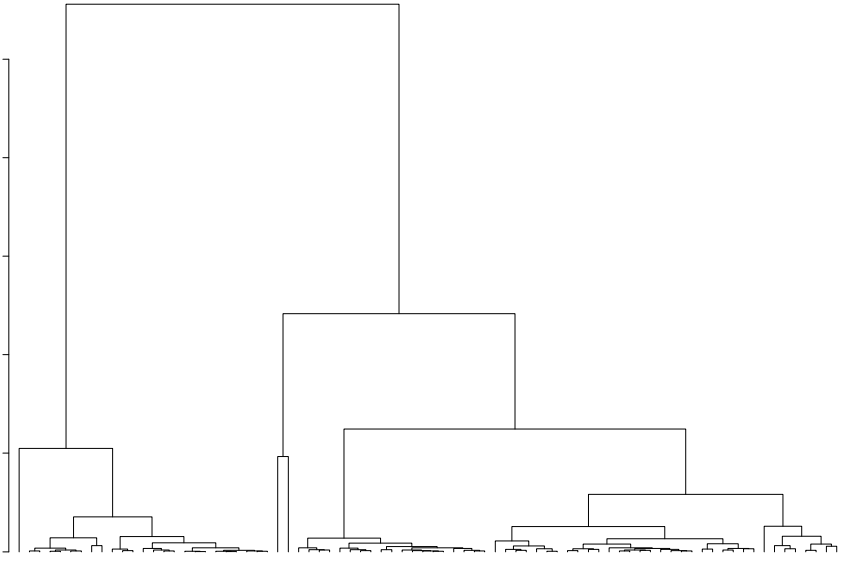


Abb. 60 Ein Dendrogramm

[♮] In Experimenten mit genetischen Algorithmen wurde beobachtet [[Kok et al. 1996]], daß eine Beschränkung auf das Fundamentalgebiet eine Konvergenzbeschleunigung für einen einzelnen Experimentlauf mit sich bringen kann. Jedoch stellte sich schnell heraus, daß die Beschränkung nicht durch Übergang zu äquivalenten Gewichtungsvektoren zustande kam, sondern durch eine Heuristik, die jedesmal, wenn der Gewichtungsvektor das Fundamentalgebiet verließ, ihn irgendwie wieder zurückbrachte; eine andere Heuristik hingegen verschlechterte die Konvergenz [[Kok 1996]].

Hierarchisches Clustern (4.5.2) Ein guter Startpunkt für die Clusteranalyse ist *hierarchisches Clustern* ⟦Duda und Hart 1973⟧: es erlaubt eine beliebige Abstandsmatrix D, z. B. $D_{ij} = d(w^i, w^j)$ und es müssen weder eine Clusteranzahl noch Clusterzentren vorgegeben werden. Zu Beginn des Clustervorgangs werden m Unterstrukturen bestehend aus je einem Gewichtungsvektor bereitgestellt. In jedem Schritt werden nun die beiden Unterstrukturen, die gemäß D den kleinsten Abstand zueinander haben, zu einer Unterstruktur zusammengefaßt. Der Abstand zweier Unterstrukturen ist dabei als *maximaler*[♮] Abstand ihrer Elemente definiert. Nach $m-1$ Schritten sind alle m Gewichtungsvektoren zu einer großen Struktur zusammengefaßt. Dendrogramme wie in Abb. 60 zeigen diesen Prozeß als einen Baum, wobei die Verzweigungshöhe den Abstand der beteiligten zwei Unterstrukturen angibt.

Cluster im Sinn dieses Kapitels werden nun durch Unterstrukturen mit kleinen Verzweigungshöhen und mindestens $\sqrt{m}$ Elementen (für eine gewisse statistische Signifikanz) gegeben. □

Visualisierung der Versuchsergebnisse (4.5.3) Nachdem ein Dendrogramm wie in Abb. 60 erzeugt wurde, liegt folgende Visualisierung der Versuchsergebnisse w^i nahe: Durch einen waagrechten Schnitt im Baum werden einige, hier drei, Cluster identifiziert; aus jedem Cluster c wird willkürlich ein Element w^{c_1} herausgegriffen und dessen Repräsentant $r(w^{c_1})$ dargestellt; alle anderen Elemente des Clusters werden als diejenigen äquivalenten Gewichtungsvektoren $\text{nearest}(w^{c_i}, r(w^{c_1}))$ gezeichnet, die $r(w^{c_1})$ am nächsten sind.

Dadurch wird eine lokal zutreffende Darstellung im effektiven Gewichtungsraum inklusive der angrenzenden Gebieten erreicht. Abb. 61 zeigt links eine Projektion der unbearbeiteten Versuchsergebnisse und rechts eine Projektion äquivalenter Gewichtungsvektoren, die – wie geschildert – die Clusterstruktur respektieren. □

♮ sog. Complete linkage method; minimaler oder mittlerer Abstand ergibt weniger kompakte Cluster

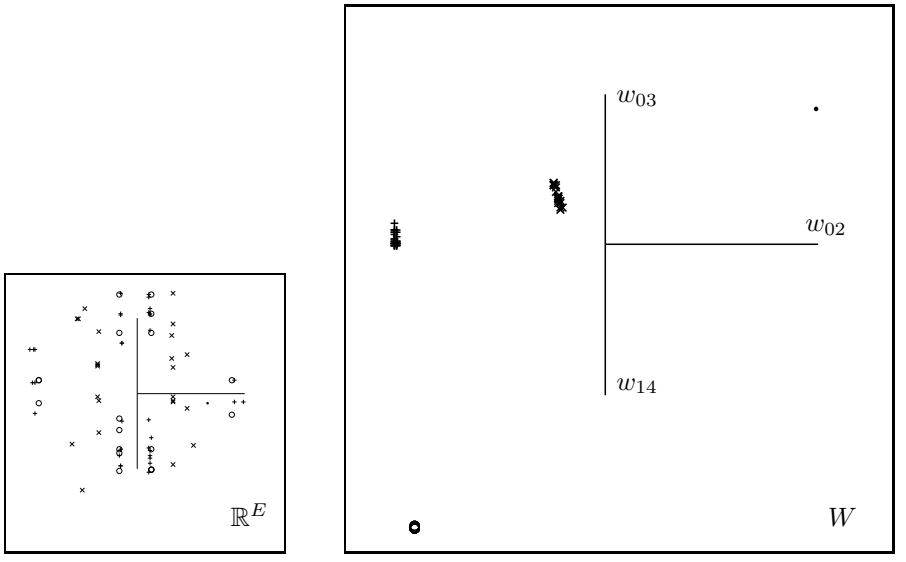


Abb. 61 Visualisierung von Gewichtungsvektoren. Links zeigen sich die unbearbeiteten Versuchsergebnisse als nicht identifizierbares Durcheinander; das Abbilden auf i. w. das Fundamentalgebiet gemäß (4.5.3) erhellt schlagartig die Clusterstruktur.

Interpretation der Clusterstruktur (4.5.4) Die Clusterstruktur der Ergebnisgewichtungsvektoren gibt deutliche Hinweise auf das typische Verhalten des Experiments. Verschiedene Cluster können z. B. durch nicht äquivalente lokale Extremwerte der Likelihood zustande gekommen sein oder nicht in Cluster fallende Ausreißer durch weitere Unzulänglichkeiten des Lernverfahrens. Schon im Sinne einer wissenschaftlich wünschenswerten Reproduzierbarkeit von Experimenten erscheint es sinnvoll, die Cluster als typische Antworten des Experiments auszuweisen. □

Diese Idee liegt folgendem, nicht beweisbaren Prinzip zugrunde.

Prinzip der identifizierbaren Antwort (4.5.5) *Wenn mehrere Läufe eines Experiments unter leicht veränderten Bedingungen keine Clusterstruktur in den Ergebnisgewichtungsvektoren zeigen, dann muß vermutet werden, daß eine der Komponenten des Experiments, wie z. B. Datenmultimenge, Lernalgorithmus oder Netzmodell, inadäquat ist.* □

4.6 Robuste Modellauswahl

In den bisherigen Kapiteln wurde bei Anwendungen künstlicher neuronaler Netze immer davon ausgegangen, daß das gewählte oder vorgegebene Netzmodell (N, E) schon dem Problem angemessen ist, also die richtige Struktur hat. Das Problem des Auffindens der *besten* Struktur gilt i. allg. als intraktabel ⟦Lin und Vitter 1989⟧. Selbst wenn die Struktur im groben festgelegt wurde (z. B. aufgrund des Theorems von Cybenko zu einer Binnenschicht), ist die Bestimmung der Zahl der Binnenknoten immer noch ein schwieriges Problem. Dies soll in diesem Abschnitt untersucht werden; dabei wird von i-h-o-Netzen die Rede sein: das sind Netze mit einer Eingabeschicht, einer Binnenschicht und einer Ausgabeschicht, mit je i, h und o Knoten, die jeweils vollständig verknüpft sind (alle Nichteingabe-Knoten haben dabei ein Biasgewicht).

Unter- und Überanpassung (4.6.1) Ein Netz mit zuwenig Binnenknoten führt zur *Unteranpassung*: die Datenmenge kann nicht angemessen repräsentiert werden, denn die möglichen Netzfunktionen sind zuwenig flexibel (Abb. 62 links). Im Gegensatz dazu können Netze mit sehr vielen Binnenknoten die Datenpaare problemlos anpassen, sogar so gut, daß die zufälligen Einflüsse mitmodelliert werden. Diese *Überanpassung* führt zu großen Schwankungen der Netzfunktion (Abb. 62 rechts). □

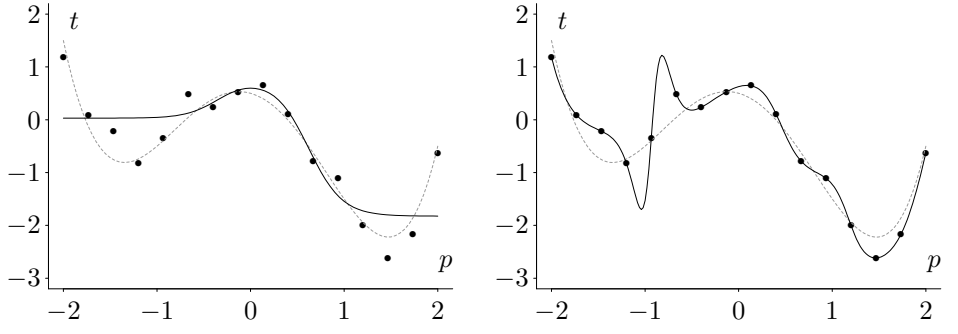


Abb. 62 Unter- und Überanpassung der Datenmenge aus Abb. 48

Generalisierung, Prognosewert und Prognosefehler (4.6.2) Normalerweise ist es nicht das Ziel neuronalen Lernens, die vorhandenen Datenpaare möglichst gut reproduzieren zu können. Es soll ein den Daten zugrundeliegender Zusammenhang aus Beispielen der Trainingsmultimenge T in einer Weise extrahiert werden, daß bei neuen, unbekannten Eingaben p, der „wahre“ Wert $\text{out}_{w^\star}(p)$ möglichst gut getroffen wird. Diese wünschenswerte Eigenschaft heißt *Generalisierung*.

Den von einer geschätzten Netzfunktion $\text{out}_{\hat{w}}$ produzierten Funktionswert $\text{out}_{\hat{w}}(p)$ nennt man auch *Prognosewert*, seine Differenz $t - \text{out}_{\hat{w}}(p)$ zum entsprechenden Wert t eines Datenpaars (p, t) den *Prognosefehler*. Der mittlere quadratische Prognosefehler $\text{E}((t - \text{out}_{\hat{w}}(p))^2)$, der sog. *Generalisierungsfehler*, ermöglicht eine quantitative Charakterisierung der Generalisierung. □

Konzeptionelle Probleme der Generalisierung (4.6.3) (i) Die Definition des Generalisierungsfehlers benutzt die Kenntnis des wahren Zusammenhangs, der in der Regel bei praktischen Anwendungen ja gerade unbekannt ist. Daher müssen im Einzelfall A-priori-Annahmen (Stetigkeit, Zugehörigkeit zu einer bestimmten Modellklasse etc.) zu Eigenschaften des wahren Zusammenhangs gemacht werden. Im allgemeinen ist es zweifelhaft – ohne solche Annahmen – eine Generalisierung über beobachtete Punkte hinaus zu wagen: es gibt dann keine eindeutige Fortsetzung des Zusammenhangs.

(ii) Zur Bildung des Erwartungswertes (in der Definition des Generalisierungsfehlers) muß eine Wahrscheinlichkeitsverteilung auf $\mathbb{R}^{I \cup O}$, dem Raum der potentiell möglichen Paare (p, t), angegeben werden. Dazu gibt es im Einzelfall u. U. keine naheliegende Wahl. Stimmt die Netzfunktion mit dem wahren Zusammenhang nämlich nicht überein, bestimmt die angenommene Wahrscheinlichkeitsverteilung auf $\mathbb{R}^{I \cup O}$ wesentlich den Wert des Generalisierungsfehlers. Trotzdem wird im folgenden davon ausgegangen, daß eine geeignete konkrete Wahl einer Wahrscheinlichkeitsverteilung erfolgt ist; es ist dann von *dem* Generalisierungsfehler die Rede.

(iii) Statistische Annahmen – wie stochastische Unabhängigkeit der Datenpaare und Einschränkung auf die Modellklasse der durch neuronale Netze

erzeugbaren Netzfunktionen – erlauben es, den Generalisierungsfehler zu schätzen. Doch häufig sind solche Schätzer selbst mit einer sehr hohen Unsicherheit behaftet, wie wir später (4.6.6) sehen werden. □

Modellauswahl (4.6.4) Eine Vielzahl von Methoden zur Bestimmung der Anzahl der freien Parameter des Modells (hier gegeben durch die Zahl der Binnenknoten) basiert auf der Schätzung und Bewertung des Generalisierungsfehlers ⟦Bishop 1995, Ripley 1996⟧: Mit Hilfe einer *Testmultimenge* von nicht zum Training verwendeten Daten wird der Generalisierungsfehler geschätzt. Nun werden Modelle mit verschiedener Anzahl von Parametern trainiert und die Schätzwerte des Generalisierungsfehlers verglichen. Schließlich wird das Modell mit dem kleinsten Schätzwert bevorzugt. □

Testfehler zur Schätzung des Generalisierungsfehlers (4.6.5) Dabei wird aus der Pattern-Target-Relation eine Testmultimenge G ausgesondert, die nicht zum Training verwendet wird. Der zum Trainingsfehler Err_T (1.2.8) analoge *Testfehler* Err_G beschreibt die Abweichung der Netzfunktion auf der Testmultimenge. □

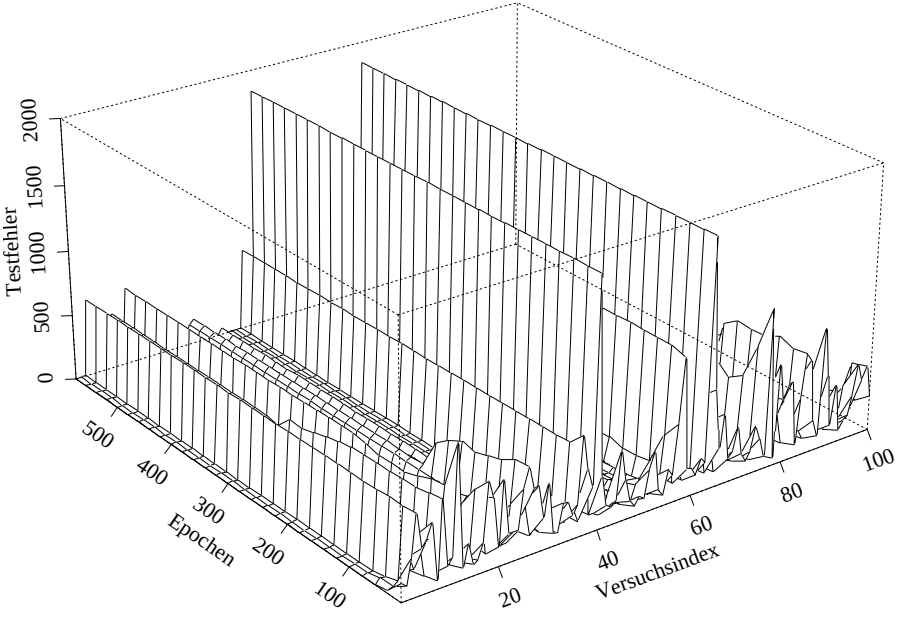


Abb. 63 Testfehlerkurven eines 3-5-1-Netzes

Experiment zur Varianz des Testfehlers (4.6.6) Folgendes Experiment dient zur Veranschaulichung der hohen Unsicherheit des Testfehlers. Es wurden 216 Datenpaare von einem 3-5-1-Netz mit einem festen Gewichtungsvektor $w^\star$ erzeugt. Die Eingabevektoren entstammen einem Gitter des Würfels $[-1,1]^3$, und die Targetvektoren wurden mit kleinen unabhängigen und identisch normalverteilten Zufallsgrößen gemäß (3.1.2) verrauscht. Dabei wurden die Daten in eine Trainings- und eine Testmenge von je 108 Paaren zufällig aufgeteilt. Mit Hilfe der Trainingsmenge wurde 100mal ein 3-5-1-Netz mit zufällig initialisierten Gewichtungsvektoren trainiert. Abb. 63 zeigt die $m = 100$ Testfehlerkurven im Verlauf des Lernens. Sogar die fertig gelernten Netze (bei Epoche 500) zeigen noch eine große Varianz des Testfehlers.

Deswegen muß eine geeignete Statistik der beobachteten Testfehler gebildet werden. Eine Ursache der großen Varianz liegt in Unzulänglichkeiten des verwendeten Lernverfahrens; hier basiert es auf Gradientenmethoden und kann daher z. B. in lokalen Minima stecken bleiben. Der Mittelwert der 100 Testfehler ist keine robuste Statistik, da sie von Ausreißern (d. h. von fehlerhaften Lernergebnissen) beeinflußt werden kann. Es bietet sich an, als robustere Statistik den Median zu verwenden. Geht man weiter davon aus, daß Ausreißer in der Regel einen zu großen Generalisierungsfehler haben werden, so ist es naheliegend, auch das Minimum als Statistik zu verwenden. Tabelle 4 zeigt in der mit $\star$ markierten Zeile diese Statistiken der 100 beobachteten Testfehler nach dem Lernen. Die sehr große relative empirische Standardabweichung von 310% demonstriert die enorme Variabilität des Testfehlers.

Dieses Experiment wurde nun für weitere Modelle mit variierender Binnenknotenzahl wiederholt. Dabei wurde je 100mal mit verschiedener Anfangsinitialisierung gelernt, nach dem Lernen der Testfehler auf der Testmenge ausgewertet, und obige Statistiken berechnet. Das Ergebnis kann in Tabelle 4 abgelesen werden.

Eine Modellauswahl, die gemäß (4.6.4) das Modell mit der kleinsten Testfehlerstatistik auswählt, würde im obigen Beispiel jeweils in einem zu komplexen Netz – dem 3-7-1-Netz für Median und Minimum, dem 3-8-1-Netz für den Mittelwert – resultieren. □

Tabelle 4 Einige Statistiken des Testfehlers

Netz	Mittelwert ± Std.	Median	Minimum
3-1-1	820,o ± 50%	622,oo	622,oo
3-2-1	330,o ± 98%	158,oo	158,oo
3-3-1	235,o ± 175%	47,6o	35,5o
3-4-1	89,o ± 240%	35,2o	3,38
⋆ 3-5-1	110,o ± 310%	3,51	1,83
3-6-1	65,o ± 420%	3,44	1,89
3-7-1	83,o ± 420%	3,18	1,68
3-8-1	9,1 ± 430%	3,31	1,84
3-9-1	13,5 ± 370%	3,33	2,20

Robuste Statistik (4.6.7) Mit der Vorarbeit von Abschnitt 4.5 ist es naheliegend, die Lernergebnisse zu clustern und die Testfehlerstatistik pro Cluster zu bilden. Zum einen muß man erwarten, daß die Testfehler innerhalb eines Clusters kaum abweichen (in der Tat stimmen im obigen Beispiel die Minimum-, Mittelwert- und Medianstatistik praktisch überein).

Zum anderen hilft Clustern bei der Interpretation der Lernergebnisse: Kann bei einem bestimmten Modell keine Clusterstruktur in den Lernergebnissen beobachtet werden, so bedeutet das, daß einige Komponenten des Gewichtungsvektors beliebige Werte annehmen können, die immer noch durch andere Komponenten kompensiert werden können. Mit anderen Worten: es liegt Überanpassung vor. Dies bekräftigt erneut das Prinzip der identifizierbaren Antwort (4.5.5).

Werden Cluster beobachtet, so kann die Kombination von Netzstruktur und typischem Gewichtungsvektor des Clusters als eigenes Modell aufgefaßt werden und zwischen diesen Modellen verglichen werden. □

Beispiel (4.6.8) Abb. 64 zeigt nun die Dendrogramme, die dem Clustern der Lernergebnisse zugrundeliegen: Die zum 3-2-1-, 3-3-1- und 3-4-1-Netz gehörigen Dendrogramme weisen große und kompakte Cluster auf. Die Clusterstruktur beginnt beim 3-5-1-Netz zu zerfallen und ist beim 3-6-1-Netz und den folgenden bereits völlig verschwunden. Obwohl die 216 Datenpaare

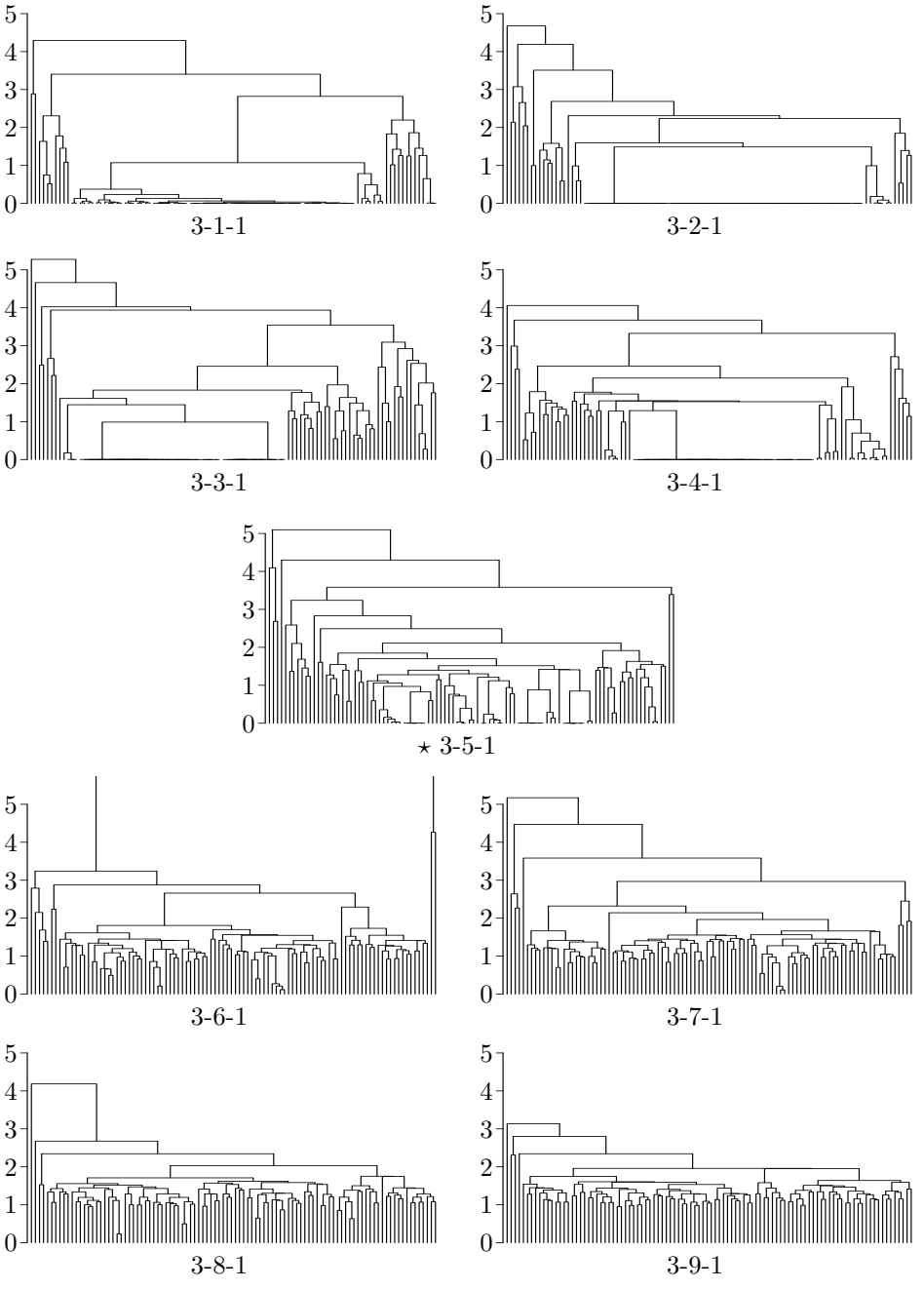


Abb. 64 Dendrogramme zur Modellauswahl

mit einem bestimmten 3-5-1-Netz erzeugt wurden, zeigt das entsprechende Dendrogramm bereits klare Anzeichen von Überanpassung. Das liegt vermutlich daran, daß nicht alle Gewichte des wahren 3-5-1-Netzes signifikant zur Netzfunktion auf dem eingeschränkten, kompakten Intervall $[-1,1]^3$ beitragen.

Tabelle 5 Pro-Cluster-Statistik des Testfehlers

Netz	Mittelwert	N/m	$N\overline{d}$
3-1-1	622,o	70%	2,7ooo
3-2-1	157,o	69%	0,0066
3-3-1	47,6	16%	0,0052
3-3-1	35,5	35%	0,0032
3-4-1	3,4	45%	0,0020
$\star$ 3-5-1	3,8	5%	0,023
$\star$ 3-5-1	2,6	5%	0,045
$\star$ 3-5-1	1,9	6%	0,017

Um die Clusterstruktur des 3-5-1-Netzes aufzulösen, wurde dieses Netz 625mal statt 100mal trainiert; es ergeben sich dann kleine Cluster. In Tabelle 5 ist die Pro-Cluster-Statistik des Testfehlers neben der relativen Clustergröße N/m und der geometrischen Größe der N-fachen durchschnittlichen Entfernung $\overline{d}$ im Cluster dargestellt. Die Pro-Cluster-Statistik würde klar ein 3-5-1-Netz mit einer bestimmten Gewichtungsbelegung (nahe $w^\star$) favorisieren, weist aber deutlich auf die Schwierigkeiten des Lernens in einem 3-5-1-Netz hin: Der „gute“ Cluster ist relativ klein, das Lernen hier dementsprechend aufwendiger. Dieselbe Tabelle legt auch nahe, das 3-4-1-Netz zu bevorzugen, wenn als Trade off ein etwas größerer Generalisierungsfehler in Kauf genommen wird. □

Vermutung (4.6.9) Tabelle 5 läßt vermuten, daß die geometrische Struktur des Clusters auch seine Güte bezüglich der Generalisierung widerspiegelt. Das ist auch deswegen naheliegend, weil gute Netzmodelle gerade dort gesucht werden müssen, wo eine noch vorhandene Clusterstruktur dabei ist, zusammenzubrechen. □

4.7 Clustered bootstrap

Ziel dieses Abschnitts ist es, die Güteeigenschaften der Schätzung der Standardabweichung der Netzfunktion durch asymptotische Methoden mit denen von Bootstrap zu vereinen. Die wesentlichen Probleme beider Verfahren lagen darin, daß bei den einzelnen Schätzungen die Unzulänglichkeiten des konkret verwendeten Algorithmus zu ungewollten Ausreißern führen. Eine naheliegende Idee ist es nun, durch Clustern im Gewichtungsraum diese Ausreißer zu eliminieren.

Clustered bootstrap (4.7.1) Sei eine Datenmultimenge T und ein bestimmtes Netzmodell gegeben (z.B. durch robuste Modellauswahl). Ziel ist es einerseits, eine T approximierende Netzfunktion $\text{out}_{\hat{w}}$ zu finden und andererseits eine Schätzung der Standardabweichung zu geben.

Zunächst werden m Maximum-Likelihood-Schätzungen mit je zufälligen Anfangsinitialisierungen durchgeführt, die resultierenden Gewichtungsvektoren geclustert und der mittlere Trainingsfehler pro Cluster bestimmt. $\hat{w}$ wird als Gewichtungsvektor mit dem besten Trainingsfehler des Clusters mit dem besten mittleren Trainingsfehler festgelegt.

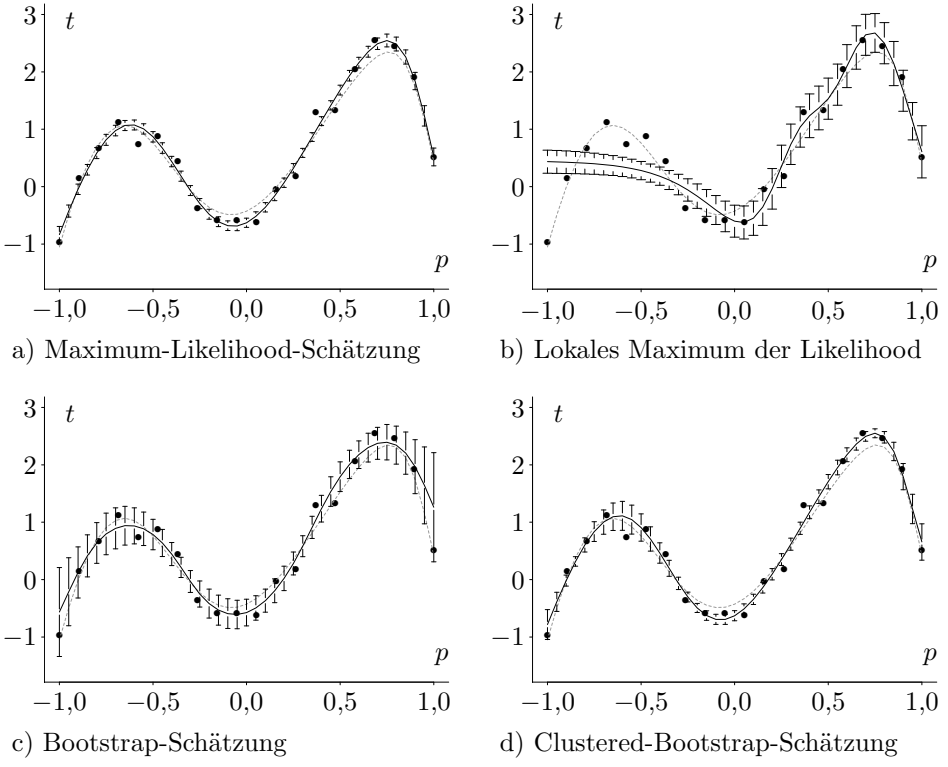
Wie bei der Bootstrap-Methode (3.4.1) zur Schätzung der Standardabweichung werden B Bootstrap-Stichproben gezogen und jeweils eine Maximum-Likelihood-Schätzung durchgeführt. $\hat{w}^{*1}, \ldots, \hat{w}^{*B}$ seien die resultierenden Gewichtungsvektoren. Zusammen mit $\hat{w}$ werden diese Gewichtungsvektoren geclustert. Der Cluster C, der $\hat{w}$ enthält, wird ausgezeichnet. Nun wird in Analogie zu (3.4.1;s) die empirische Standardabweichung

$$\sqrt{\frac{1}{|C^*|-1}\sum_{w\in C^*}(\text{out}_w(p)-\frac{1}{|C^*|}\sum_{w\in C^*}\text{out}_w(p))^2}$$

der Netzfunktionen eingeschränkt auf $C^* := C \setminus \{\hat{w}\}$ als Schätzer für die Standardabweichung der Netzfunktion vorgeschlagen. □

Vergleich am Beispiel (4.7.2) Seinen 20 Datenpaare mit je eindimensionaler Eingabe und Ausgabe gegeben und sei durch eine Modellauswahl bereits ein 1-3-1-Netz festgelegt worden. In Abb. 65 sind einige Techniken

dargestellt, Schätzungen des den 20 Datenpaaren zugrunde liegenden Zusammenhangs zu beurteilen. Jeweils sind die Datenpaare als dicker Punkt, der „wahre Zusammenhang“ als unterbrochene Linie und die Schätzung des Zusammenhangs als durchgezogene Linie eingezeichnet. Die Variabilität der Schätzung ist als symmetrisches Intervall dargestellt, das nach oben und unten die geschätzte Standardabweichung abdeckt.



a) Maximum-Likelihood-Schätzung

b) Lokales Maximum der Likelihood

c) Bootstrap-Schätzung

d) Clustered-Bootstrap-Schätzung

Abb. 65 Zusammenfassung der Beurteilung von Schätzungen

Abb. 65a zeigt die Netzfunktion der Maximum-Likelihood-Schätzung, deren Variabilität mit der Delta-Methode bestimmt wurde. Die Schätzung hat diverse Güteeigenschaften: sie ist konsistent, asymptotisch erwartungstreu, asymptotisch effizient und asymptotisch normalverteilt. Letzteres ist die theoretische Grundlage der Schätzung der Variabilität der Netzfunktion. Leider zeigen typisch angewendete Verfahren zur Bestimmung des Maximums der Likelihood Unzulänglichkeiten: Abb. 65b demonstriert, was

passiert, wenn nur ein lokales Maximum der Likelihood gefunden wird. Hier gibt das Ergebnis die Daten nicht nur falsch wieder, sondern täuscht auch eine hohe Zuverlässigkeit vor.

Die rechenintensive Bootstrap-Methode (Abb. 65c) spiegelt in der mittleren Bootstraplinie und den großen Fehlerbalken auch die Defizite des konkreten Schätzverfahrens wider, ist dadurch andererseits aber auch nicht genügend genau. Ein Mangel, der schließlich von Clustered bootstrap aufgehoben wird (Abb. 65d): Das Verfahren ist genauso robust wie die Bootstrap-Methode, berechnet aber viel genauer die Variabilität der Netzfunktion. □

Bemerkung (4.7.3) Die *Ortsinformation* des Maximum-Likelihood-Gewichtungsvektors $\hat{w}$ der Gesamtdaten T geht bei Clustered bootstrap wesentlich ein: Es wird über den Cluster von Gewichtungsvektoren gemittelt, in dem sich $\hat{w}$ befindet. Es hätte keinesfalls gereicht, die Likelihoodwerte $L_{T^{*i}}(w^{*i})$ der Bootstrap-Gewichtungsvektoren w^{*i} zu clustern. Die Likelihoodfunktionen $L_{T^{*i}}$ sind nämlich *verschiedene* Funktionen, deren Funktionswerte nicht notwendigerweise vergleichbar sind. Experimente ergeben, daß gute Bootstrap-Likelihoodwerte durch unzureichende Bootstrap-Gewichtungsvektoren zustande kommen können und umgekehrt. Das Vorhandensein von Clustern zeigt zudem an, daß keine Überanpassung der Daten vorliegt. □

Verbesserung von Prognosedichten (4.7.4) Anstatt den Prognosewert zusammen mit der Standardabweichung für ein Eingabemuster p zu schätzen, kann mit Hilfe der empirischen Bootstrap-Verteilung der Netzausgaben, die durch $i \mapsto \mathrm{out}_{w^{*i}}(p)$ gegeben wird und einem Kernschätzer eine Prognosedichte geschätzt werden. Dies ist zweifellos ein Vorteil der Bootstrap-Methode. Abb. 66 (entnommen aus ⟦Ossen und Rüger 1996b⟧) zeigt zwei Beispiele; rechts ist der gleiche Fall wie links dargestellt, nur wurde die empirische Bootstrap-Verteilung auf den besten Cluster im Sinne

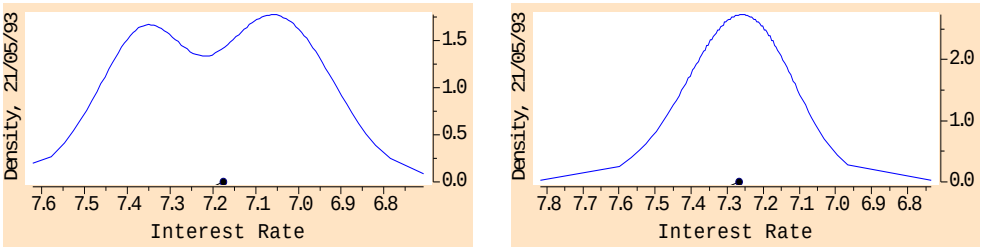


Abb. 66 Prognosedichten

von Clustered bootstrap eingeschränkt. Es wurden also wieder gezielt Prognosewerte weggelassen, die durch unzulängliche Schätzungen zustande kamen. Die rechte Kurve ist im Gegensatz zur linken unimodal, was den theoretischen Erwartungen entgegenkommt, und schmaler, was einer genaueren Angabe der Prognoseunsicherheit entspricht. □

4.8 Wertung

Ziel dieses Kapitels war es, unzureichende Lernergebnisse mit Hilfe der *geometrischen Lageinformation* von Gewichtungsvektoren zu identifizieren. Dies wurde zunächst erschwert durch in vielfältiger Weise im Gewichtungsraum wirkendende Symmetrien. Die Analyse der zugrundeliegenden relevanten Symmetriegruppe erlaubte die Identifikation eines nichtredundanten Teilbereichs, das sog. Fundamentalgebiet, des Gewichtungsraums.

Sollen Gewichtungsvektoren, die eine ähnliche Bedeutung – d. h. ähnliche Netzfunktion – haben, auf den ersten Blick als ähnlich erkannt werden, so reicht es nicht, sie im Fundamentalgebiet darzustellen. Vielmehr mußte ein geeigneter Abstandsbegriff eigens in diesem Kapitel eingeführt werden. Dieser kann dann auch zu einer Clusteranalyse der Gewichtungsvektoren mehrerer Versuchsläufe verwendet werden.

Der dem Clustern zugrundeliegende Abstandsbegriff entspringt in natürlicher Weise theoretischen Überlegungen. Der kanonische Abstand zwischen Gewichtungsvektoren kann zwar nicht in traktabler Weise exakt berechnet werden, ist jedoch in gleicher Weise wie das Problem des Handlungsreisenden effizient und ausreichend genau approximierbar. Durch die zum Clustern notwendige Anzahl der Versuchsläufe sind sämtliche Methoden

dieses Kapitels rechenintensiv. Sie bieten dafür die Möglichkeit, Gewichtungsvektoren zu eliminieren, die durch Unzulänglichkeiten des Lernverfahrens zustande kommen. Zugleich können die durch Wiederholung der Experimente gewonnenen Erkenntnisse zu einer soliden Beurteilung der Versuchsergebnisse herangezogen werden.

Die Ortsinformation eines Clusters mit gutem Trainingsfehler kann zur Identifizierung des geeigneten Clusters von Bootstrap-Gewichtungsvektoren benutzt werden. Dies führt zu einer präziseren Einschätzung des Prognosefehlers. Das wiederum verbessert die Beurteilung des Risikos, das mit einer Entscheidung aufgrund der Prognosewerte gekoppelt ist.

Das Clustern im Gewichtungsraum erlaubt eine Prüfung einer gesamten Anwendung bestehend aus Datensatz, Netzmodell und Lernalgorithmus aufgrund des in diesem Kapitel eingeführten *Prinzips der identifizierbaren Antwort*. Wird diese Prüfung bestanden, zeigt die Anwendung also konsistente Lernergebnisse in Clustern, heißt das noch nicht, daß die Anwendung erfolgreich sein wird. Aber die Gewichtungsvektoren in den Clustern können als Ausgangspunkt zum Bilden einer robusten Statistik des Testfehlers, und damit zur Beurteilung des Netzmodells, herangezogen werden. Bewahrheitet sich, daß die geometrische Struktur der Cluster Rückschlüsse auf die Generalisierung erlaubt, könnte ein Algorithmus konstruiert werden, der ähnlich einer Autofokusvorrichtung in einer Kamera auf ein geeignetes Netzmodell fokussiert. Der auf der Hand liegende Vorteil wäre, daß eine Testmenge nicht mehr benötigt wird: die volle Information kann zum Trainieren verwendet werden.

Die hier vorgestellten Methoden lassen sich darüber hinaus nicht nur für eine Beurteilung von Netzmodellen, sondern in naheliegender Weise auch für eine objektivere Beurteilung von Lernverfahren benutzen, z. B. in der Angabe der relativen Clustergröße der erfolgreichen Lernvorgänge.

5 Die Boltzmann-Maschine

$$S = k \ln \Phi$$

— Inschrift des Grabsteins von
L. Boltzmann nach ⟦Resnikoff 1989⟧

Self-reference is a precondition for life, not a problem. Each living cell carries the plans for itself.

— D. Finkelstein in ⟦Finkelstein 1988⟧

5.1 Die traditionelle Boltzmann-Maschine

Die Boltzmann-Maschine ⟦Hinton und Sejnowski 1983⟧ ist das erste stochastische Netz, für das eine Lernregel ⟦Ackley et al. 1985, Hinton und Sejnowski 1986⟧ angegeben wurde. Die Lernregel basiert auf lokalen Wechselwirkungen der Knoten, wodurch sie parallel implementierbar ist. Die theoretischen Grundlagen der Boltzmann-Maschine stammen von der statistischen Physik, was eine gewisse Relevanz zusichert.

Definition (5.1.1) Eine *Boltzmann-Maschine* (N, E, w, T) besteht

- aus einem ungerichteten Graphen (N, E), d. i. eine Menge N von Knoten und eine nicht leere symmetrische Menge $E \subset N \times N$ von Kanten mit $(a, b) \in E \Rightarrow (b, a) \in E$,
- ohne *Selbstwechselwirkung*, d. h. $a \in N \Rightarrow (a, a) \notin E$,
- mit einer Partitionierung von N in S (sichtbare Knoten), U (unsichtbare oder auch Binnenknoten genannt) und $\{0\}$ (Biasknoten),
- symmetrischen Gewichten $w\colon E \to \mathbb{R}$ ($w_{ab} = w_{ba}$ für alle $(a, b) \in E$)
- und stochastischen Bipolar-Aktivationen $s\colon (U \cup S) \to \{-1, 1\}$

$$s_a = \begin{cases} +1 & \text{mit Wahrscheinlichkeit } \sigma(\frac{2}{T} \sum_{b \in R_a} w_{ab} s_b) \\ -1 & \text{ansonsten,} \end{cases}$$

wobei der spezielle Knoten 0 (der Biasknoten) immer die Aktivation $s_o = 1$ haben soll. Die *Updates* der Aktivationen erfolgen *asynchron*: In einem Zeitschritt wird ein Knoten gleichverteilt zufällig ausgewählt und entsprechend obiger Update-Regel die Aktivation des Knotens zufällig bestimmt.

In diesem Sinn ist s ein Zufallsvektor und die Boltzmann-Maschine ein Markow-Prozeß; s heißt auch *Zustand* des Netzes. Der Parameter $T \in \mathbb{R}^+$ heißt *Temperatur*; σ ist die Fermifunktion $x \mapsto 1/(1 + \exp(-x))$. □

Beispiel (5.1.2) In Abb. 67 ist eine sehr kleine Boltzmann-Maschine mit einem unsichtbaren Knoten und den zu den Kanten assoziierten Gewichten gezeichnet. *Dieses Netz löst bei kleinen Temperaturen das Xor-Problem, d. h. leistet die Zuordnung* $(-1,-1) \mapsto -1$, $(-1,1) \mapsto 1$, $(1,-1) \mapsto 1$ *und* $(1,1) \mapsto -1$. □

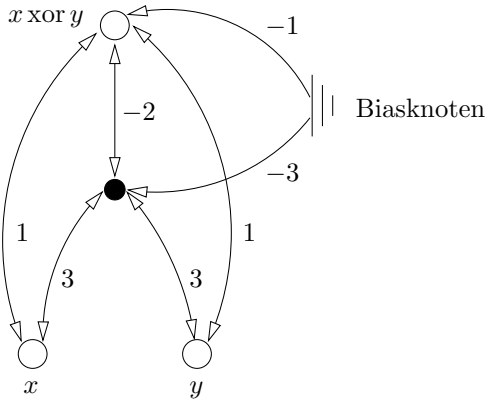


Abb. 67 Beispiel einer Boltzmann-Maschine für das Xor-Problem

Beweis (5.1.3) Betrachte die Abbildung $x \mapsto \sigma(x/T)$ aus Abb. 68. Bei sehr kleinen Temperaturen wird diese Abbildung der Sprungfunktion θ immer ähnlicher: für alle $x \neq 0$ gilt nämlich

$$\lim_{T \to 0} \sigma(\frac{x}{T}) = \theta(x).$$

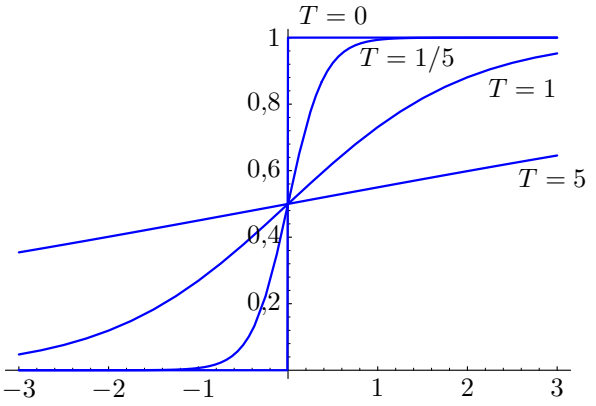


Abb. 68 Die Abbildung $x \mapsto \sigma(x/T)$

Daher ist für kleine Temperaturen die Update-Regel aus (5.1.1) im wesentlichen, d. h. für von Null verschiedene Argumente, gegeben durch

$$s_a = \operatorname{sign}\Big(\sum_{b \in R_a} w_{ab} s_b\Big).$$

Durch einfaches Einsetzen wird nachgewiesen, daß die Aktivation am unsichtbaren Knoten • durch die logische Und-Verknüpfung von x und y gegeben ist (unabhängig von der Aktivation des Ausgabeknotens ∘). Die Aktivationen x und y werden dabei von außen vorgegeben. Nachdem nun einmal der unsichtbare Knoten einen Update erfahren hat, wird beim nächsten Update der Ausgabeknoten ∘ die Xor-Funktion berechnen. Nach einer kurzen Einschwingzeit (bis das erste Mal der unsichtbare Knoten und danach der Ausgabeknoten der Update-Regel unterzogen wurde) ist die Boltzmann-Maschine stabil. ∎

Bedeutung der Gewichte (5.1.4) Wird die Wahrscheinlichkeit für Aktivation 1 bei einem Update eines Knotens aus (5.1.1) mit Trennung des Einflusses des Biasknotens aufgeschrieben, das ist

$$\sigma\Big(\frac{2}{T}\Big(\sum_{b \in R_a, b \neq 0} w_{ab} s_b + w_{ao}\Big)\Big),$$

so ist deutlich ersichtlich, daß das Biasgewicht w_{ao} den Erwartungswert von s_a beeinflußt und im Prinzip dadurch festgelegt werden kann. Genauso ist ersichtlich, daß das Gewicht w_{ab} den Erwartungswert von $s_a s_b$

und damit die Korrelation von s_a zu s_b beeinflußt. Korrelationen höherer Ordnung (z. B. beim Xor-Problem) können dann indirekt über Gewichte zu Binnenknoten kodiert werden. □

Definition und Satz (5.1.5) *Die Energie* $H\colon \mathbb{R}^E \times \{-1,1\}^N \to \mathbb{R}$

$$H(w,s) := -\frac{1}{2} \sum_{(a,b)\in E} w_{ab} s_a s_b$$

wird für $T = 0$ *minimiert, sobald ein stabiler Zustand erreicht ist.* □

Beweis (5.1.6) Angenommen, es tritt ein Wechsel der Aktivation eines Knotens c von s_c zu s'_c um $\Delta s_c = 2s'_c$ ein. Bei der Energie können die von c abhängigen Terme getrennt aufgeschrieben werden:

$$H(w,s) = -\frac{1}{2} \sum_{\substack{(a,b)\in E \\ a\neq c \wedge b \neq c}} w_{ab} s_a s_b - 2 \cdot \frac{1}{2} \sum_{a\in L_c} w_{ac} s_a s_c$$

Dabei wurde sowohl die fehlende Selbstwechselwirkung als auch die Symmetrie der Gewichte durch $\sum_{a\in L_c} w_{ac} s_a s_c = \sum_{a\in R_c} w_{ca} s_c s_a$ berücksichtigt. Dann ändert sich die Energie um

$$\Delta H_c = -\Delta s_c \sum_{a\in L_c} w_{ac} s_a,$$

die sog. *Flip-Energie*. Kam nun der Wechsel der Aktivation eines Knotens c durch die Regel $s'_c = \operatorname{sign}(\sum_a w_{ac} s_a)$ zustande, so ist $\Delta H_c \leq 0$. Nur wenn das Argument $\sum_a w_{ac} s_a$ gleich null ist, ergibt sich keine Energieänderung. Kann umgekehrt die Update-Regel keine Änderung des Zustands s bewirken, so ist die Energie minimal bezüglich der *Nachbarzustände*, die sich in nur einer Komponente von s unterscheiden. ■

Gleichgewicht (5.1.7) Sei $N_* := N \setminus \{0\}$ die Menge der Knoten ohne den Biasknoten. Es gibt $2^{|N_*|}$ verschiedene Zustände aus $\{-1,1\}^{N_*}$ (diese Angabe reicht, denn der Biasknoten 0 hat immer Aktivation 1). Die Übergangswahrscheinlichkeit von einem Zustand in einen anderen kann in einer Matrix $Q\colon \{-1,1\}^{N_*} \times \{-1,1\}^{N_*} \to \mathbb{R}$ notiert werden. Die Übergangswahrscheinlichkeit ist null, falls s und s' sich in mehr als einem Knoten unterscheiden. Sind s und s' Nachbarzustände, die sich in einem Knoten

unterscheiden, so ist $Q_{s\,s'}$ die sog. *Flip-Wahrscheinlichkeit* $\sigma\big(\big(H(w,s) - H(w,s')\big)/T\big)$ aus Bemerkung (5.1.8;i) geteilt durch $|N_*|$.

Sei nun q^o ein Zeilenvektor, der die initiale Wahrscheinlichkeitsverteilung der Zustände angibt, d. h. jedem möglichen Zustand $s \in \{-1,1\}^{N_*}$ seine Wahrscheinlichkeit zuordnet. Dann gibt q^oQ die Wahrscheinlichkeitsverteilung der Zustände nach einem Zeitschritt, q^oQQ die nach zwei Zeitschritten und q^oQ^t die nach der Zeit t an. Das System hat eine *stabile Wahrscheinlichkeitsverteilung* p, wenn

$$p = pQ,$$

also p ein Eigenvektor der Matrix Q zum Eigenwert 1 ist.

Ist umgekehrt bei endlich vielen Zuständen nach einiger Zeit t jedes Element von Q^t strikt positiv (von einem beliebigen Anfangszustand heraus kann jeder Zustand mit positiver Wahrscheinlichkeit zur Zeit $t+1$ erreicht werden), so gibt es eine eindeutig bestimmte stabile Wahrscheinlichkeitsverteilung, die obiges Eigenwertproblem löst und unabhängig von s' gilt

$$p_s = \lim_{t\to\infty} (Q^t)_{s'\,s}.$$

Die Matrix Q^t existiert für $t \to \infty$ und enthält in ihren identischen Zeilen die stabile Wahrscheinlichkeitsverteilung ⟦Krengel 1991, § 15⟧.

Für die Boltzmann-Maschine ist dieser Sachverhalt aber bei Temperatur $T > 0$ gerade gegeben: Zu jedem beliebigen Anfangszustand s gibt es nach $|N_*|$ Zeitschritten eine positive Wahrscheinlichkeit, einen beliebigen Zustand s' zu erreichen; es kann nämlich sein, daß pro Zeitschritt der Reihe nach immer ein anderer der Knoten aus N_* der Update-Regel unterzogen werden soll und daß das Zufallsexperiment am Knoten a gerade die Aktivation s'_a zum Ergebnis hat.

Anders formuliert: Die Zustände sind Ecken eines Hyperwürfels und in einem Zeitschritt kann ein Zustand zu irgend einem benachbarten (durch eine Kante des Hyperwürfels verbundenen) Zustand gelangen, nach zwei Zeitschritten zu irgendeinem Zustand, der über eine oder zwei Kanten mit dem ursprünglichen verbunden ist und nach $|N_*|$ Zeitschritten kann jeder Zustand mit positiver Wahrscheinlichkeit (von einem beliebigen Anfangszustand heraus) erreicht sein.

Nach einigen Zeitschritten nähert sich die Wahrscheinlichkeitsverteilung der Zustände einer Boltzmann-Maschine der stabilen Wahrscheinlichkeitsverteilung; man sagt auch, die Boltzmann-Maschine befindet sich im *(thermischen) Gleichgewicht*. Dadurch ist zwar kein bestimmter Zustand gegeben und die Zustände werden im Gleichgewicht fluktuieren, jedoch ist deren Wahrscheinlichkeitsverteilung stabil.

Der *Erwartungswert* einer Funktion X der Zustände ist im Gleichgewicht gegeben durch

$$\mathrm{E}(X) := \sum_{s \in \{-1,1\}^N} p(s) X(s) \qquad \square$$

Bemerkungen (5.1.8) (i) Die Update-Regel aus (5.1.1) ist äquivalent damit, die Aktivation des Knotens a mit der Flip-Wahrscheinlichkeit

$$\mathrm{Prob}(s_a \to -s_a) = \sigma(-\Delta H_a / T)$$

zu ändern, wobei ΔH_a die Flip-Energie aus (5.1.6) ist.

(ii) Die Matrix Q ist eine *stochastische Matrix* (Elemente nichtnegativ und Zeilensumme 1), d. h.

$$Q_{s\,s'} \geq 0 \quad (s, s' \in N_*) \quad \text{und} \quad \sum_{s' \in N_*} Q_{s\,s'} = 1 \quad (s \in N_*). \qquad \square$$

Boltzmann-Gibbs-Verteilung (5.1.9) Die stabile Wahrscheinlichkeitsverteilung einer Boltzmann-Maschine ist durch die sog. *Boltzmann-Gibbs-Verteilung* gegeben,

$$P_w := s \mapsto \frac{\exp(-H(w,s)/T)}{Z_w},$$

wobei Z_w ein Normierungsfaktor (die sog. *Zustandssumme*) ist mit

$$Z_w = \sum_{s \in \{-1,1\}^N} \exp(-H(w,s)/T).$$

Die Boltzmann-Gibbs-Verteilung ist für ein ähnliches Problem aus dem kanonischen Formalismus der Thermodynamik bekannt ⟦Callen 1985⟧.

Sie läßt sich aber auch im obigen Rahmen sehr gut motivieren: Das Gleichgewicht durch die stabile Wahrscheinlichkeitsverteilung p kann z. B. dadurch erreicht werden, daß die mittlere Zahl der Übergänge von Nachbarzuständen s nach s' mit der von s' nach s übereinstimmt:

$$p_s Q_{s\,s'} = p_{s'} Q_{s'\,s}$$

Es folgt daraus mit der Abkürzung $\Delta = \big(H(w,s') - H(w,s)\big)/T$

$$\begin{aligned}\frac{p_s}{p_{s'}} = \frac{Q_{s'\,s}}{Q_{s\,s'}} &= \frac{\sigma(\Delta)/|N_*|}{\sigma(-\Delta)/|N_*|} \\ &= \frac{1+\exp(\Delta)}{1+\exp(-\Delta)} \\ &= \frac{\exp(\Delta)(\exp(-\Delta)+1)}{1+\exp(-\Delta)} \\ &= \frac{\exp(-H(w,s)/T)}{\exp(-H(w,s')/T)}.\end{aligned}$$

Damit ist bis auf die Normierung die stabile Wahrscheinlichkeitsverteilung der Zustände festgelegt und es ergibt sich die Boltzmann-Gibbs-Verteilung. □

Bemerkung (5.1.10) Die gewählte Update-Regel aus (5.1.1) ist nicht die einzige Möglichkeit, obige Gleichgewichtsbedingung zu erfüllen. Zum Beispiel führt die Festlegung der Flip-Wahrscheinlichkeit

$$\mathrm{Prob}(s_a \to -s_a) = \begin{cases} 1 & \text{falls } \Delta H_a < 0 \\ \exp\big(-\Delta H_a/T\big) & \text{sonst} \end{cases}$$

genauso zur Boltzmann-Gibbs-Verteilung. Diese Wahl ist als *Metropolis-Algorithmus* ⟦Metropolis et al. 1953⟧ bekannt und führt schneller zum Gleichgewicht, da im Mittel mehr Zustandswechsel passieren. □

Anwendungsgebiet der Boltzmann-Maschine (5.1.11) Die Boltzmann-Maschine als stochastisches Netz erlaubt es, Wahrscheinlichkeitsverteilungen an den sichtbaren Knoten zu approximieren. Das kann benutzt werden für

- Modellierung stochastischer Vorgänge,
- Schließen unter Ungewißheit,

- Musterergänzung (*Pattern completion*) oder um
- bekanntes I/O-Verhalten zu modellieren.

Das Xor-Problem aus (5.1.2) kann angesehen werden als das Lernen der Wahrscheinlichkeitsverteilung, die jedem Vektor aus $\{---, -++, +-+, ++-\}$ die Wahrscheinlichkeit $1/4$ und allen anderen aus $\{-,+\}^3$ die Wahrscheinlichkeit 0 zuordnet. Eine Boltzmann-Maschine, die dieses Problem gelernt hat, kann an einem Teil der sichtbaren Knoten die resultierenden Aktivationen ergänzen, wenn an den restlichen sichtbaren Knoten ein Muster fest vorgegeben ist (Pattern completion).

Ein Beispiel für die Modellierung stochastischer Vorgänge ist das Nachbilden des Auftretens bestimmter Symptome bei einer Krankheit. Die Ausprägung eines Symptoms bei diagnostizierter Krankheit ist zwar bestimmt kein stochastischer Prozeß; dem Arzt aber, dem die innere biochemische und psychische Struktur seiner Patienten nicht bis zur letzten Wirkung bewußt ist, muß die mal mehr und mal weniger starke Ausprägung eines Symptoms als zufällig erscheinen. Bei noch nicht diagnostizierter Ursache gibt ihm seine statistische Erfahrung Hinweise auf mögliche Krankheitsursachen, die er mit Hilfe gezielter Untersuchungen unterscheiden kann. Eine Boltzmann-Maschine könnte die in Abb. 69 gezeigte Wahrscheinlichkeitsdichte des Auftretens der Symptome S_1 und S_2 nachgebildet haben und ebenso anzeigen, ob und welche Differentialdiagnose notwendig ist. □

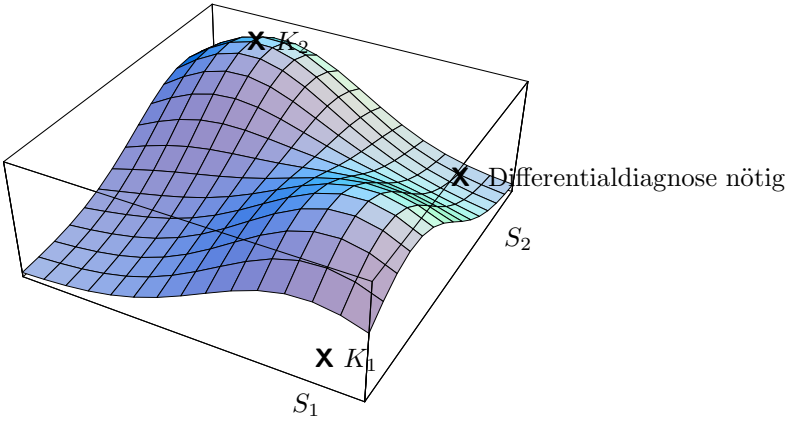


Abb. 69 Wahrscheinlichkeitsdichte für die Symptome S_1 und S_2

5.2 Lernregel für die Boltzmann-Maschine

Ein im folgenden noch zu lösendes Problem ist es, den Gewichtungssatz w so einzustellen, daß an den sichtbaren Knoten eine bestimmte Wahrscheinlichkeitsverteilung approximiert wird.

Relative Entropie (5.2.1) Sei $\alpha \in \{-1,1\}^S$ ein Zustand der sichtbaren Knoten, $\beta \in \{-1,1\}^U$ ein Zustand der unsichtbaren Knoten; dann ist der Zustand $s := \alpha\beta \in \{-1,1\}^{S\cup U}$ der Boltzmann-Maschine dadurch vollständig gegeben. Die im Gleichgewicht gegebene Wahrscheinlichkeitsverteilung an den sichtbaren Knoten ist eine sog. *Randverteilung* p_w der Boltzmann-Gibbs-Verteilung mit

$$p_w(\alpha) = \sum_\beta P_w(\alpha\beta);$$

sie hängt von den Gewichten ab. Die gewünschte Wahrscheinlichkeitsverteilung heiße r.

Ein geeignetes Maß für den informationstheoretischen Unterschied zwischen zwei Wahrscheinlichkeitsverteilungen r und p ist durch die sog. *relative Entropie*

$$E_w(r,p) = -\sum_\alpha r(\alpha)\log\frac{p(\alpha)}{r(\alpha)}$$

gegeben ⟦Kullback 1959⟧. *E_w ist nichtnegativ und nur null für $r = p$.* □

Beweis (5.2.2) $x \mapsto -\log(x)$ ist eine konvexe Funktion. Durch Anwendung der Jensenschen Ungleichung (5.2.3) auf E_w entsteht sofort die behauptete Ungleichung $E_w(r,p) \geq -\log(\sum p(\alpha)) = 0$. Die Gleichheit wird angenommen für $r = p$; anderenfalls hat $r(\alpha)/p(\alpha)$ eine nichttriviale Verteilung, was wegen der strikten Konvexität von $-\log$ dazu führt, daß die Gleichheit der Jensenschen Ungleichung nicht angenommen werden kann. ■

Jensensche Ungleichung (5.2.3) Eine Abbildung $q\colon I \to \mathbb{R}$ auf einem nicht leeren Intervall $I \subset \mathbb{R}$ heißt *konvex*, wenn für alle $x, y \in I$ und alle $0 \leq \lambda \leq 1$ gilt

$$q(\lambda x + (1-\lambda)y) \leq \lambda q(x) + (1-\lambda)q(y),$$

m. a. W. wenn der Graph der Funktion q zwischen den Punkten $(x, q(x))$ und $(y, q(y))$ unterhalb oder auf der Verbindungslinie der beiden Punkte liegt. Die *Jensensche Ungleichung* ⟦Bauer 1991, Satz 3.9⟧ besagt nun für reellwertige Zufallsvariable X, die in ein offenes Intervall I abbilden und deren Erwartungswert $\mathrm{E}(X)$ existiert, daß mit gegebenem konvexen $q\colon I \to \mathbb{R}$ folgt

$$q(\mathrm{E}(X)) \leq \mathrm{E}(q \circ X). \qquad \square$$

Ableitung einer Lernregel (5.2.4) Minimiere die Abbildung $w \mapsto E_w(r, p_w)$ mit der Methode des Gradientenabstiegs:

$$\begin{aligned}
\Delta w_{ab} :&= w_{ab}^{i} - w_{ab}^{i-1} \\
&= -\eta \frac{\partial\big(w \mapsto E_w(r, p_w)\big)}{\partial w_{ab}} \\
&= \eta \sum_{\alpha} \frac{r(\alpha)}{p_w(\alpha)} \frac{\partial p_w(\alpha)}{\partial w_{ab}}
\end{aligned}$$

Nun ist

$$\begin{aligned}
\frac{\partial p_w(\alpha)}{\partial w_{ab}} &= \sum_{\beta} \frac{\partial P_w(\alpha\beta)}{\partial w_{ab}} \\
&= \frac{\sum_\beta \exp\big(-H(w,s)/T\big) s_a s_b / 2T}{Z_w} \\
&\quad - \frac{\sum_\beta \exp\big(-H(w,s)/T\big) \cdot \sum_s \exp\big(-H(w,s)/T\big) s_a s_b / 2T}{Z_w \cdot Z_w} \\
&= \frac{1}{2T}\Bigg(\sum_{\beta} P_w(\alpha\beta) s_a s_b - p_w(\alpha)\, \mathrm{E}(s_a s_b) \Bigg),
\end{aligned}$$

und eingesetzt in Δw_{ab} ergibt sich

$$\begin{aligned}
\Delta w_{ab} &= \frac{\eta}{2T}\Bigg(\sum_{\alpha} r(\alpha) \sum_{\beta} \frac{P_w(\alpha\beta)}{p_w(\alpha)} s_a s_b - \underbrace{\sum_{\alpha} r(\alpha)}_{1}\, \mathrm{E}(s_a s_b) \Bigg) \\
&= \frac{\eta}{2T}\Big(\overline{\mathrm{E}(s_a s_b)}_{\alpha}^{r} - \mathrm{E}(s_a s_b) \Big). \qquad (L)
\end{aligned}$$

Die letzte Gleichung (L) ist als *Boltzmann-Lernregel* bekannt. Je nachdem, ob r in einem Modell als Wahrscheinlichkeitsverteilung oder durch Beispiele als empirische Verteilung vorgegeben ist, bedeute $\overline{X}^r$ den Mittelwert oder den Erwartungswert von X bezüglich r. Des weiteren sei die *Wechselwirkung* $\mathrm{E}(s_a s_b)_\alpha$ definiert als der Erwartungswert von $s_a s_b$ bezüglich der bedingten Wahrscheinlichkeit $P_w(\beta|\alpha) := P_w(\alpha\beta)/p_w(\alpha)$ dafür, daß an den unsichtbaren Knoten der Zustand β anliegt, wenn α fest vorgegeben ist. Man sagt auch, das Muster α wäre *aufgeprägt*. Die Wechselwirkung $\mathrm{E}(s_a s_b)$ hingegen sei, wie in (5.1.7) definiert, der Erwartungswert über die Boltzmann-Gibbs-Verteilung. □

Anwendung der Lernregel (5.2.5) Die Erwartungswerte $\mathrm{E}(\bullet)$ und $\mathrm{E}(\bullet)_\alpha$ können zwar im Prinzip berechnet werden, jedoch erfordert schon allein die Berechnung der Zustandssumme die Bestimmung von $2^{|N|}$ Summanden. Dies ist schon bei kleiner Knotenzahl nicht mehr praktikabel.

Der Ergodensatz, eine Verallgemeinerung des Gesetzes der großen Zahl ⟦Krengel 1991, § 11⟧, erlaubt es jedoch, den Erwartungswert $\mathrm{E}(s_a s_b)$ als zeitliches Mittel von $s_a s_b$ im Gleichgewicht zu nähern. Dies hat den enormen Vorteil, daß eine lokale Beobachtung der beiden betroffenen Aktivationen s_a und s_b reicht, um den Erwartungswert ihres Produkts zu bestimmen. Genauso kann für jedes α mit $r(\alpha) > 0$ der Erwartungswert $\mathrm{E}(s_a s_b)_\alpha$ bestimmt werden: Das Muster α wird an die sichtbaren Knoten angelegt, die Boltzmann-Maschine jedesmal ins Gleichgewicht gebracht und danach das zeitliche Mittel von $s_a s_b$ gemessen. Sind sowohl a als auch b sichtbare Knoten, so kann die Mittelung natürlich entfallen, denn $s_a s_b$ ist dann schon durch das Muster α vorgegeben.

Nun sind kleine Temperaturen nötig für eine schnelle Anpassung der Verteilung p_w an die gewünschte Verteilung r (besonders wenn r wie im Xor-Beispiel deterministisch ist); gerade bei kleinen Temperaturen kann es jedoch sehr lange dauern bis die Boltzmann-Maschine von alleine ins Gleichgewicht kommt. Es kann jedoch ein Verfahren, das sog. *Simulated annealing* angewendet werden ⟦Metropolis et al. 1953, Hecht-Nielsen 1989⟧, um die Boltzmann-Maschine schneller ins Gleichgewicht zu bringen. Die Idee und der Name des Verfahrens stammen aus dem metallurgischen

Vergütungsprozeß; dabei wird ein Metall stark (z. B. bis zum Schmelzpunkt) erhitzt und danach langsam abgekühlt. Bei hoher Temperatur verliert das Metall seine Kristallgitterdefekte; beim langsamen Abkühlen kann es dann in das Energieminimum der perfekten Kristallgitteranordnung gelangen.

Ein möglicher konkreter Algorithmus für dieses Verfahren („Lasse zunächst die Boltzmann-Maschine bei hoher Temperatur arbeiten und senke danach die Temperatur langsam bis zur gewünschten Arbeitstemperatur T.“) ist im folgenden formuliert. □

Simulated annealing (5.2.6) Sei (N, E, w, T_o) eine Boltzmann-Maschine mit Arbeitstemperatur T_o und $s \in \{-1,1\}^N$ ihr Zustand. Typische Werte von T_o liegen in der Größenordnung von $|N|$ oder darunter. Für ein fest vorgegebenes Muster $\alpha \in \{-1,1\}^S$ soll die Boltzmann-Maschine in ihr thermisches Gleichgewicht gebracht werden (setze $S = \emptyset$, wenn kein von außen aufgeprägtes Muster erwünscht ist).

Variablen: $s \in \{-1,1\}^N$, $H^o, H^a, H^t, T \in \mathbb{R}$ und $n, m, t \in \mathbb{N}_o$.

Parameter des Algorithmus (*Annealing schedule*): $\lambda \in [\frac{3}{4}, 1)$, $\mu < \nu \in \mathbb{N}$, $\varepsilon \ll T_o$ und $\tau \gg \nu$; die Parameter λ, μ und ν bestimmen die Geschwindigkeit des Abkühlens und ε und τ sollen helfen, ein frühzeitig eingestelltes Gleichgewicht zu erkennen.

(i) Initialisierung: Wähle ein großes T ($\geq 10|N| \max_{ab}\{|w_{ab}|\}$). Setze $m \leftarrow 0$ und $n \leftarrow 0$. Setze $s_a \leftarrow \alpha_a$ für alle $a \in S$. Setze $s_a \in \{-1,1\}$ zufällig für alle $a \in U$. Berechne $H_o \leftarrow -\sum_{(a,b)\in E} w_{ab}s_as_b/2$ und setze $H^t \leftarrow H^o$ und $H^a \leftarrow H^o$.

(ii) Update eines Knotens

- Suche zufällig einen Knoten k aus U mit Gleichverteilung aus.
- Berechne
$$\Delta H_k = \underbrace{-\Delta s_k}_{2s_k}\Big(\sum_{b\in R_k} w_{kb}s_b\Big).$$
- Ändere die Aktivation s_k zu $-s_k$ mit der Flip-Wahrscheinlichkeit $\min\big(1, \exp(-\Delta H_k/T)\big)$ des Metropolis-Algorithmus, d. h. ziehe eine

gleichverteilte Zufallszahl $r \in [0,1)$, vergleiche sie mit der Flip-Wahrscheinlichkeit und ändere die Aktivation, wenn r kleiner war.

- Wurde die Aktivation geändert, so setze $H^o \leftarrow H^o + \Delta H_k$, inkrementiere n und, falls $\Delta H_k < 0$, inkrementiere m.
- Setze $H^a \leftarrow (19H^a + H^o)/20$ (gleitendes Mittel). Inkrementiere t falls $|H^a - H^t| < \varepsilon$, ansonsten setze $t \leftarrow 0$ und $H^t \leftarrow H^a$.

(iii) Abkühlung: Sind μ erfolgreiche, d. h. energieverkleinernde, oder ν Änderungen insgesamt einer Aktivation erfolgt ($m \geq \mu \vee n \geq \nu$), so setze $T \leftarrow \lambda T, n \leftarrow 0$ und $m \leftarrow 0$.

(iv) Stoppe, wenn die Arbeitstemperatur erreicht ist ($T_o \geq T$) oder wenn H^o nahezu unverändert seit den letzten τ Knoten-Updates ist ($t \geq \tau$). Anderenfalls gehe zu (ii). □

5.3 Variationen der Boltzmann-Maschine

Weight decay (5.3.1) Gewichte mit kleinem Betrag sind auch bei der Boltzmann-Maschine von Vorteil. Wird nach jeder Anwendung der Boltzmann-Lernregel (5.2.4;L) mit einem kleinen $\varepsilon \in \mathbb{R}^+$

$$w_{ab} \leftarrow (1 - \varepsilon) w_{ab}$$

gesetzt, so zerfallen nicht wirklich benötigte Gewichte zu Null. Da die Lernregel symmetrisch in a und b ist, werden bei dieser Methode des *Weight decay* die Gewichte im Lauf der Zeit symmetrisch, auch wenn sie zu Anfang nicht so gewählt wurden. □

Verrauschte Eingabemuster (5.3.2) Falls nur wenige Muster $\alpha \in \{-1,1\}^S$ eine gewünschte Wahrscheinlichkeit $r(\alpha) > 0$ haben, muß die Boltzmann-Maschine oft die Wahrscheinlichkeit 0 approximieren. Dies geht nur exakt, wenn einige Gewichte unendlich hohe Werte haben, was aber die Lerndauer ungünstig beeinflußt. Dies wird vermieden, wenn beim Lernen statt des Musters α ein leicht verrauschtes Muster (mit einer bestimmten Wahrscheinlichkeit wird jede Aktivation gekippt) angelegt wird. □

Definition und Satz: Eingabe-Ausgabe-Maschine (5.3.3) *Ein Zustand $\alpha\gamma$ der sichtbaren Knoten bestehe aus einem Zustand α der Eingabeknoten und einem Zustand γ der Ausgabeknoten mit einer Eingabe-Ausgabe-Relation, die dazwischen vermittelt. Sei – wie vorher – r die gewünschte Verteilung der sichtbaren Knoten und p_w die durch die Gewichte an der Boltzmann-Maschine eingestellte. Dies führt mit $q = \alpha \mapsto \sum_\gamma r(\alpha\gamma)$ als Randverteilung der Eingabemuster zu der Lernregel*

$$\Delta w_{ab} = \frac{\eta}{2T}\left(\overline{\mathrm{E}(s_a s_b)}^{r}_{\alpha\gamma} - \overline{\mathrm{E}(s_a s_b)}^{q}_{\alpha}\right)$$

bezüglich der Kostenfunktion $w \mapsto \overline{E_w(r(\gamma|\alpha), p_w(\gamma|\alpha))}^{q}$. □

Beweis (5.3.4) $r(\gamma|\alpha) = r(\alpha\gamma)/q(\alpha)$ ist die bedingte Wahrscheinlichkeit dafür, daß eine bestimmte Ausgabe anliegen soll, gegeben eine Eingabe. Daher spaltet sich die relative Entropie $E_w(r, p_w)$ auf in

$$E_w(r, p_w) = \sum_\alpha q(\alpha) \log \frac{q(\alpha)}{\sum_\gamma p_w(\alpha\gamma)} + \sum_{\alpha\gamma} q(\alpha) r(\gamma|\alpha) \log \frac{r(\gamma|\alpha)}{p_w(\gamma|\alpha)}.$$

Der erste Summand ist die relative Entropie der Randwahrscheinlichkeiten. Bei der Eingabe-Ausgabe-Maschine soll die Wahrscheinlichkeit des Auftretens einer bestimmten Eingabe gar nicht modelliert werden. Deswegen beschränkt man sich auf den zweiten Summanden, den *Information gain* I_w. Er gibt – gemittelt über alle Eingaben α – an, wie weit die von der Eingabe-Ausgabe-Maschine gegebene bedingte Wahrscheinlichkeitsverteilung $\gamma \mapsto p_w(\gamma|\alpha)$ mit der gewünschten $\gamma \mapsto r(\gamma|\alpha)$ übereinstimmt.

Nun ist die bedingte Verteilung

$$p_w(\gamma|\alpha) = \frac{p_w(\alpha\gamma)}{\sum_\gamma p_w(\alpha\gamma)} = \frac{\sum_\beta P_w(\alpha\beta\gamma)}{\sum_{\beta\gamma} P_w(\alpha\beta\gamma)} = \frac{\sum_\beta \exp(-H(w, \alpha\beta\gamma)/T)}{\sum_{\beta\gamma} \exp(-H(w, \alpha\beta\gamma)/T)},$$

und deren Ableitung bestimmt sich mit $s = \alpha\beta\gamma$ zu

$$\begin{aligned}\frac{\partial p_w(\gamma|\alpha)}{\partial w_{ab}} =& \frac{\sum_\beta \exp(-H(w,s)/T) s_a s_b}{2T \sum_{\beta\gamma} \exp(-H(w,s)/T)} \\ &- \frac{\sum_\beta \exp(-H(w,s)/T) \cdot \sum_{\beta\gamma} \exp(-H(w,s)/T) s_a s_b}{2T\Big(\sum_{\beta\gamma} \exp(-H(w,s)/T)\Big)^2}.\end{aligned}$$

Dies eingesetzt in

$$\Delta w_{ab} = -\eta \frac{\partial I_w}{\partial w_{ab}} = \eta \sum_{\alpha} q(\alpha) \sum_{\gamma} \frac{r(\gamma|\alpha)}{p_w(\gamma|\alpha)} \frac{\partial p_w(\gamma|\alpha)}{\partial w_{ab}}$$

ergibt sogleich das gewünschte Resultat für Δw_{ab}

$$\frac{\eta}{2T}\Big(\sum_{\alpha\gamma} r(\alpha\gamma) \sum_{\beta} P_w(\beta|\alpha\gamma) s_a s_b - \sum_{\alpha} q(\alpha) \sum_{\beta\gamma} P_w(\beta\gamma|\alpha) s_a s_b \Big). \qquad \blacksquare$$

Hopfield-Netze (5.3.5) Das *Hopfield-Netz* (N, w) kann angesehen werden als eine spezielle Boltzmann-Maschine bei Temperatur 0 ohne Binnenknoten, bei der jeder Knoten mit jedem verbunden ist. Es dient der autoassoziativen Speicherung von n als Bipolarvektoren kodierten Bildern $\xi^1, \xi^2, \ldots, \xi^n \in \{-1, 1\}^{N_*}$. Die asynchron anzuwendende Update-Regel für Hopfield-Netze lautet

$$s_a = \operatorname{sign}\big(\sum_{b \in N} w_{ab} s_b \big).$$

Da keine Binnenknoten vorhanden sind, vereinfacht sich die Boltzmann-Lernregel (5.2.4;L) bei Annahme einer Gleichverteilung der Bilder zu

$$\Delta w_{ab} = \eta \left(\frac{1}{n} \sum_{i=1}^{n} \xi_a^i \xi_b^i - \mathrm{E}(s_a s_b) \right).$$

Der erste Summand ist eine von den Bildern abhängige Konstante. Es liegt daher nahe, die Gewichte schon durch die – als Hebbsche Lernregel[♮] bekannte – Gleichung

$$w_{ab} = \frac{1}{n} \sum_{i=1}^{n} \xi_a^i \xi_b^i$$

zu initialisieren. Dies reicht schon für einige beeindruckende Experimente: Ein paar am besten nicht untereinander korrelierte Bilder können so gespeichert werden. Zum Abrufen eines bestimmten Bildes reicht es, nur einen Teil davon aufzuprägen und die Aktivationen der restlichen Knoten durch

[♮] Der Psychologe D. O. Hebb hatte Ende der 40iger Jahre postuliert, daß die Änderung der Synapsenstärke („Gewichte“) proportional zur Korrelation der Feuerraten („Aktivationen“) ihrer prä- und postsynaptischen Neuronen („Knoten“) ist ⟦Hebb 1949, Hertz et al. 1991⟧.

die Update-Regel solange bestimmen zu lassen, bis der Zustand s sich nicht mehr ändert. Ebenso kann ein verrauschtes Bild wiederhergestellt werden: Es wird als initialer Zustand verwendet, und die Update-Regel bezieht alle Knoten ein.

Statt der (später aufgestellten) Boltzmann-Lernregel $\Delta w_{ab} = -\varepsilon\,\mathrm{E}(s_a s_b)$ für die wie oben initialisierten Gewichte wurde vorgeschlagen, von zufälligen Zuständen aus zu starten, das Hopfield-Netz laufen zu lassen bis ein stabiler Zustand s^* erreicht ist, der nach (5.1.5) zu einem lokalen Minimum der Energie gehört, und dann

$$\Delta w_{ab} = -\varepsilon s_a^* s_b^*$$

zu bilden ⟦Hopfield et al. 1983⟧. Es wurde demonstriert, daß dieses sog. *Hebbsche Vergessen* eine große Kapazitätserweiterung (Anzahl von Bildern, die gespeichert werden können) mit sich bringt und automatisch damit auch korrelierte Bilder gespeichert werden können ⟦van Hemmen et al. 1990⟧. □

Deterministische Boltzmann-Maschine (5.3.6) Dies ist die vielleicht bekannteste Variation der Boltzmann-Maschine. Dabei wird zum einen der Ansatz gemacht, daß Aktivationen nahezu stochastisch unabhängig sind,

$$\mathrm{E}(s_a s_b) \approx \mathrm{E}(s_a)\,\mathrm{E}(s_b),$$

wobei sich der Erwartungswert von s_a zu

$$\begin{aligned}\mathrm{E}(s_a) &= \mathrm{Prob}(s_a = 1)\cdot 1 + \mathrm{Prob}(s_a = -1)\cdot(-1)\\ &= 2\sigma\big(\frac{2}{T}\sum_{b\in R_a} w_{ab}s_b\big) - 1\\ &= \tanh\big(\frac{1}{T}\sum_{b\in R_a} w_{ab}s_b\big)\end{aligned}$$

berechnet. In einer weiteren Näherung, im sog. *Mean-field-Ansatz*, wird das fluktuierende „Feld“ $\sum_b w_{ab}s_b$ durch das zeitlich gemittelte, also vom aktuellen Zustand s unabhängige, „Feld“ $\sum_b w_{ab}\,\mathrm{E}(s_b)$ ersetzt. Dies führt zu einem Gleichungssystem

$$\mathrm{E}(s_a) = \tanh\big(\frac{1}{T}\sum_{b\in R_a} w_{ab}\,\mathrm{E}(s_b)\big)$$

von $|N_*|$ nichtlinearen Gleichungen mit $|N_*|$ Unbekannten $\mathrm{E}(s_\bullet)$. Eine numerische Lösungsmethode besteht z. B. darin, die Iteration

$$\mathrm{E}(s_a)^{\mathrm{neu}} \leftarrow \tanh\big(\frac{1}{T}\sum_{b\in R_a} w_{ab}\,\mathrm{E}(s_b)^{\mathrm{alt}}\big)$$

bis zu einem Fixpunkt durchzuführen. Auch hier mag ein allmähliches Absenken der Temperatur helfen, schneller zur Lösung des Gleichungssystems zu gelangen. Die Lernregel für die deterministische Boltzmann-Maschine reduziert sich auf folgende Rechnung:

$$\Delta w_{ab} = \frac{\eta}{2T}\left(\overline{\mathrm{E}(s_a)_\alpha\,\mathrm{E}(s_b)_\alpha}^{\,r} - \mathrm{E}(s_a)\,\mathrm{E}(s_b)\right)$$

Die Gewichte werden verändert ohne die zufälligen Aktivationen der Boltzmann-Maschine zu simulieren. Die Erfinder ⟦Peterson und Anderson 1987⟧ dieser Methode gaben Beispiele an, in denen diese Lernregel 10–30 mal schneller war als die Boltzmann-Lernregel (5.2.4;L); dabei erzielten sie sogar etwas bessere Resultate. Die Annahme der stochastischen Unabhängigkeit der Aktivationen ist jedoch sehr einschränkend und es ist zu erwarten, daß stark korrelierte Zusammenhänge schlechter gelernt werden. □

5.4 Schließen in Boltzmann-Maschinen

Nachdem die Boltzmann-Maschine trainiert ist, beinhaltet sie implizit – durch Beispiele gewonnene – statistische Zusammenhänge. Diese gezielt abzufragen, ist Gegenstand des Schließens in Boltzmann-Maschinen.

Schließen in Boltzmann-Maschinen (5.4.1) Beim Abfragen einer Boltzmann-Maschine (N, E, w, T) wird ihr sämtliches vorhandene Wissen – hier der Zustand bestimmter Knoten – aufgeprägt. Die Menge der Knoten mit aufgeprägter Eingabe sei I. Sodann wird die Menge O der interessierenden Knoten festgelegt. Schließen in einer Boltzmann-Maschine bedeute nun das Berechnen der bedingten Wahrscheinlichkeit von $\gamma \in \{-1,1\}^O$ gegeben $\alpha \in \{-1,1\}^I$. Diese ist von zentraler Bedeutung für alle in (5.1.11) auftauchenden Anwendungen. Es wird hier die Eingabe-Ausgabe-Maschine (5.3.3) betrachtet.

Die Mengen I, O und $U := N \setminus (O \cup I \cup \{0\})$ sind dabei nicht notwendigerweise dieselben wie beim Trainieren der Boltzmann-Maschine und können von Anwendungsfall zu Anwendungsfall verschieden sein. Wurde aber mit einer Eingabemenge I_t trainiert, so sollte jedenfalls $I \supset I_t$ sein, denn die Kostenfunktion des Information gain beinhaltet nicht die Modellierung der Auftrittswahrscheinlichkeiten von Mustern in I_t. Dies ist insofern keine Einschränkung, als daß I_t für das Training beliebig klein – also auch als leere Menge – gewählt werden konnte. □

Schließen in Boltzmann-Maschinen ist NP-hart (5.4.2) Sei (N, E, w, T) eine Boltzmann-Maschine mit der Boltzmann-Gibbs-Verteilung P_w. Die beim Schließen interessierende bedingte Wahrscheinlichkeit

$$p_w(\gamma|\alpha) = \sum_{\beta \in \{-1,1\}^U} P_w(\beta\gamma|\alpha)$$

für $\alpha \in I$ und $\gamma \in O$ erfordert formell $2^{|U|}$ Summationen, selbst wenn $P_w(\beta\gamma|\alpha)$ effizient berechnet werden kann.

Für verwandte Netze – die sog. Belief-Netze (6.6.1) – wurde nachgewiesen, daß das Schließen in ihnen NP-hart ist ⟦Cooper 1990⟧, ja sogar daß die Approximation der bedingten Wahrscheinlichkeit NP-hart ist ⟦Dagnum und Luby 1993⟧.

Nun bedeutet die Tatsache, daß sowohl bei Belief-Netzen als auch bei Boltzmann-Maschinen Knoten und Kanten zur Beschreibung verwendet werden, nicht, daß es die gleichen Dinge sind. Der Beweis in ⟦Cooper 1990⟧ läßt sich aber übertragen und soll im folgenden skizziert werden. □

Beweisidee (5.4.3) Ziel ist es zu zeigen, daß die Erfüllbarkeit des sog. 3SAT-Problems durch Schließen in einer Boltzmann-Maschine berechnet werden kann, wobei die zum Problem gehörige Boltzmann-Maschine mit polynomiellem Aufwand konstruiert werden kann.

Beim 3SAT-Problem hat man eine Menge $C = \{c_1, \ldots, c_m\}$ von Boolschen Ausdrücken und eine Menge $U = \{u_1, \ldots, u_n\}$ von Boolschen Variablen. Jeder Boolsche Ausdruck ist dabei eine Disjunktion dreier Variabler oder negierter Variabler aus U, also z.B. $c_1 = u_2 \vee \neg u_4 \vee \neg u_1$ (daher die 3 im Namen 3SAT). Die Beantwortung der Frage, ob alle Boolschen Ausdrücke

in C gleichzeitig erfüllt werden können, ist nun bezüglich n NP-hart. Einen viel besseren Algorithmus als das Prüfen aller 2^n Belegungen der Variablen $u_1, \ldots, u_n$ scheint es nach dem Stand der Dinge nicht zu geben.

Sei nun eine Boltzmann-Maschine zu C und U derart gegeben, daß die n Binnenknoten $u_1, \ldots, u_n$ eine zufällige Belegung haben, und daß andere m Knoten $c_1, \ldots, c_m$ die entsprechenden Boolschen Ausdrücke berechnen. Die nötigen Boolschen Schaltelemente können etwa wie beim Xor-Beispiel in Abb. 67 aussehen. Die Zeit zur Konstruktion der Boltzmann-Maschine ist linear in den Parametern n und m.

Durch Schließen in der Boltzmann-Maschine kann die von der Temperatur T abhängige gemeinsame Wahrscheinlichkeit dafür berechnet werden, daß alle Knoten $c_1, \ldots, c_m$ positive Aktivation haben (d. h. alle Ausdrücke in C gemeinsam erfüllt sind). Ist diese Wahrscheinlichkeit beim Übergang $T \to 0$ immer noch positiv, so muß das zugrundeliegende 3SAT-Problem mit ja, ansonsten mit nein beantwortet werden. Mithin konnte das 3SAT-Problem mit polynomiellem Aufwand auf das Schließen in einer Boltzmann-Maschine transformiert werden. □

5.5 Ausblick

In diesem Kapitel wurde zunächst eine Darstellung der aus der Literatur bekannten Boltzmann-Maschine gegeben und danach argumentiert, warum das Schließen in ihnen i. allg. NP-hart ist.

Zu wissen, daß Schließen in Boltzmann-Maschinen NP-hart ist, ist außerordentlich nützlich: bedeutet es doch, daß mit sehr geringer Priorität ein effizienter Algorithmus zum Schließen in allgemeinen Boltzmann-Maschinen gesucht werden sollte. Statt dessen sollte vielmehr in Richtung von effizienten Algorithmen für geeignete – genügend interessante – Spezialfälle von Boltzmann-Maschinen geforscht werden. Das ist die Motivation dafür, die Klasse der im nächsten Kapitel definierten *dezimierbaren Boltzmann-Maschinen* zu studieren.

6 Dezimierbare Boltzmann-Maschinen

Not far from here, by a white sun, behind a green star, lived the Steelypips, illustrious, industrious, and they hadn't a care: no spats in their vats, no rules, no schools, no gloom, no evil influence of the moon, no trouble from matter or antimatter — for they had a machine, a dream of a machine, with springs and gears and perfect in every respect. And they lived with it, and on it, and under it, and inside it, for it was all they had — first they saved up all their atoms, then they put them all together, and if one didn't fit, why they chipped at it a bit, and everything was just fine ...

— Stanisław Lem, „Cyberiad"

Im allgemeinen ist die Berechnung der lokalen Wechselwirkung zwischen den Knoten einer Boltzmann-Maschine intraktabel bezüglich der Anzahl der Binnenknoten. Mit Hilfe des Ergodensatzes und Simulated annealing können die Wechselwirkungen jedoch ausreichend gut geschätzt werden. Viele Anstrengungen, wie z. B. die deterministische Boltzmann-Maschine, sind seither unternommen worden, um diese – selbst für heutige Computer oder spezielle Hardware [[Alspector et al. 1992, Murray et al. 1994]] – sehr langsame Technik zu verbessern.

Eine besonders interessante Methode ist Dezimierung [[Eggarter 1974, Itzykson und Drouffe 1991, Saul und Jordan 1994]]: Die Wechselwirkung zweier Knoten kann exakt berechnet werden, indem der Rest des Netzes so verkleinert wird, daß die Wechselwirkung davon unberührt bleibt. In beliebig komplexen Netzen ist das nicht möglich; die von Saul und Jordan eingeführte Methode funktioniert aber z. B. in baumartigen Strukturen. Dort ist der Berechnungsaufwand der Wechselwirkung polynomiell in der Anzahl der Knoten.

Im ersten Abschnitt dieses Kapitels wird die zugrundeliegende Idee der Dezimierung abstrahiert und so erweitert, daß auch viel komplexere Graphen dezimiert werden können. Dieser deutliche Fortschritt kann – zumindest mit Dezimierung – nicht weiter ausgebaut werden: im folgenden wird gezeigt, daß es keine weiteren Dezimierungsregeln mehr geben kann.

Durch die Fourier-Stieltjes-Transformation der Boltzmann-Gibbs-Verteilung konnte ich Lernen und Schließen in dezimierbaren Boltzmann-Maschinen auf leicht zu implementierende Algorithmen mit effizienter Ausführungszeit zurückführen [[Rüger 1996c]]. Dazu habe ich zunächst einen fundamentalen Zusammenhang zwischen der Fourier-Stieltjes-Transformierten und der Zustandssumme einer Boltzmann-Gibbs-Verteilung aufgezeigt. Trotz aller Bemühung läßt sich die Darstellung an dieser Stelle nicht so anschaulich in Bilder fassen, wie das bisher geschehen ist. Die Anstrengung des Lesens wird jedoch durch das Ergebnis belohnt: Die stochastische und rekurrente dezimierbare Boltzmann-Maschine kann nämlich in ein deterministisches, nicht rekurrentes Abbildungsnetz umgewandelt werden. Die Kettenregel in Abbildungsnetzen aus Kapitel 1 kann hier zum *gleichzeitigen* Berechnen aller lokaler Wechselwirkungen benutzt werden.

Die Boltzmann-Lernregel basiert auf der Methode des Gradientenabstiegs. Um die Verbesserungsvorschläge aus dem zweiten Kapitel nutzen zu können, muß neben dem Gradienten der Kostenfunktion auch die Kostenfunktion selbst berechnet werden können. Es wird gezeigt, wie dies in dezimierbaren Boltzmann-Maschine in traktabler Weise gelingt.

6.1 Dezimierung

Konvention (6.1.1) In diesem Kapitel bedeute v den durch die Temperatur geteilten *effektiven Gewichtungssatz* $v = w/T$. □

Ansatz der Dezimierung (6.1.2) Die Idee bei der Dezimierung ist, einen Knoten, sagen wir 1, mit all seinen Gewichten wegzulassen und statt dessen zwischen den Knoten, mit denen 1 Verbindung hatte, Kanten mit Gewichten einzusetzen (siehe Abb. 70). Diese neuen Gewichte sollen solche Werte bekommen, daß sie nun den Teil der Wechselwirkung des restlichen

Netzes, der vorher über Knotens 1 realisiert wurde, äquivalent wiedergeben. Es ergibt sich die mathematische Forderung, daß die Randverteilung der Boltzmann-Gibbs-Verteilung des ursprünglichen Systems bezüglich s_1 mit der Boltzmann-Gibbs-Verteilung des neuen Systems übereinstimmt. Am Beispiel der Abb. 70 ergibt sich

$$\sum_{s_1} P^{\mathrm{a}}_{vT}(s_1, s_2, s_3, \ldots) = P^{\mathrm{b}}_{v'T}(s_2, s_3, \ldots). \qquad \square$$

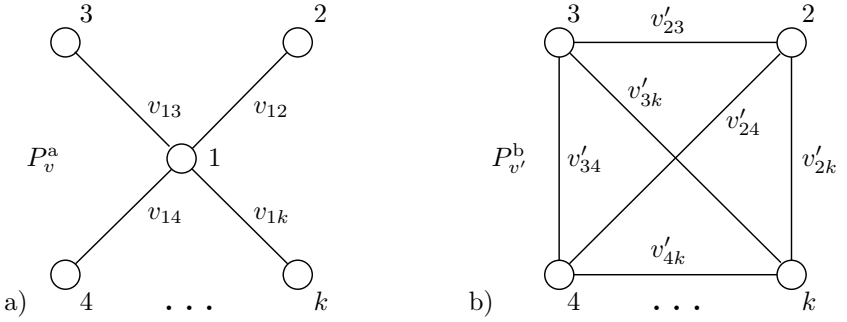


Abb. 70 Ansatz zur Dezimierung des Knotens 1; Satz (6.1.3;iii) zeigt, daß der Ansatz nur lösbar ist für $k \leq 4$ – d. h. Knoten 1 darf höchstens drei Nachbarn haben

Satz: Dezimierung (6.1.3) *$1, \ldots, k, \ldots, n$ $(1 \leq k \leq n)$ seien die Knoten[♮] einer Boltzmann-Maschine mit Arbeitstemperatur $T \neq 0$ und effektivem Gewichtungssatz $v = w/T$. Knoten 1 sei bipolar (d. h. nicht Biasknoten und $s_1 \in \{-1, 1\}$) und nur mit den $k-1$ Knoten 2, …, k verbunden. Die Knoten 2, …, n seinen bipolar bis auf einen, den Biasknoten.*
(i) Für $k = 4$ läßt sich Ansatz (6.1.2) auflösen zu

$$v'_{23} = \log\left(\frac{\cosh(v_{12}+v_{13}-v_{14})\cosh(v_{12}+v_{13}+v_{14})}{\cosh(v_{12}-v_{13}+v_{14})\cosh(v_{12}-v_{13}-v_{14})}\right)/4$$

$$v'_{24} = \log\left(\frac{\cosh(v_{12}-v_{13}+v_{14})\cosh(v_{12}+v_{13}+v_{14})}{\cosh(v_{12}+v_{13}-v_{14})\cosh(v_{12}-v_{13}-v_{14})}\right)/4$$

$$v'_{34} = \log\left(\frac{\cosh(v_{12}+v_{13}+v_{14})\cosh(v_{12}-v_{13}-v_{14})}{\cosh(v_{12}+v_{13}-v_{14})\cosh(v_{12}-v_{13}+v_{14})}\right)/4.$$

[♮] Statt einer Aufzählung $a_1, \ldots, a_n$ der Knoten werden zur Vermeidung von Doppelindizes konkrete Namen verwendet; der Biasknoten heißt hier nicht 0.

(ii) $k = 3$ *ist ein Spezialfall hiervon; durch Setzen von* $v_{14} = 0$ *im obigen Gleichungssystem ergibt sich eine entsprechende Lösung. Für* $k \leq 2$ *kann der Knoten* 1*, ggf. mit seiner einzigen Kante, weggelassen werden.*

(iii) Für $k > 4$ *hat Ansatz (6.1.2) i. allg. keine Lösung.*

(iv) Ein neues Gewicht v'_{ab} *aus (i) und (ii) muß gegebenenfalls zu einem bereits bestehenden Gewicht* v_{ab} *und zum symmetrischen Gewicht* v_{ba} *addiert werden (Parallelenregel). Gab es keine Kante* (a, b)*, so sind Kanten* (a, b) *und* (b, a) *mit jeweils dem Gewicht* v'_{ab} *einzufügen.* □

Beweis (6.1.4) (iv) Die Energie (5.1.5) ist die zentrale definierende Größe einer Boltzmann-Maschine; sie bestimmt die Boltzmann-Gibbs-Verteilung. Die Parallelenregel ist eine einfache Konsequenz aus der Linearität der Energie in den effektiven Gewichten: Die beiden in Abb. 72b gezeigten Netze haben gleiche Energie.

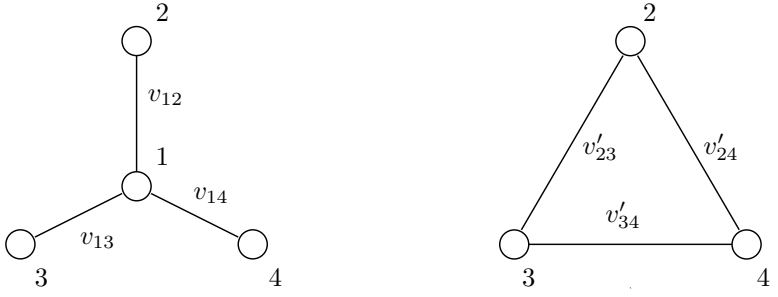


Abb. 71 Sterndezimierung des Knotens 1

(i) Durch Dezimierung des Knotens 1 wird beim Übergang von der Boltzmann-Maschine in Abb. 70a zu der von Abb. 70b auch die Zustandssumme geändert, die ja in subtiler Weise von den effektiven Gewichten abhängt. Um diese Abhängigkeiten nicht betrachten zu müssen, können im Ansatz (6.1.2) relative Wahrscheinlichkeiten

$$\frac{\sum_{s_1} P^{\mathrm{a}}_{vT}(s_1, s_2, s_3, \ldots)}{\sum_{s_1} P^{\mathrm{a}}_{vT}(s_1, 1, 1, \ldots)} = \frac{P^{\mathrm{b}}_{v'T}(s_2, s_3, \ldots)}{P^{\mathrm{b}}_{v'T}(1, 1, \ldots)}$$

benutzt werden. Der jeweilige Nenner $P^{\mathrm{b}}_{w'}(1, 1, \ldots) = \sum_{s_1} P^{\mathrm{a}}_{w}(s_1, 1, 1, \ldots)$ verschwindet nicht bei endlichen Gewichten und einer Temperatur $T \neq 0$.

Es ergibt sich dann

$$\frac{\cosh\left(\sum_{a=2}^{4} v_{1a}\right)}{\cosh\left(\sum_{a=2}^{4} v_{1a}s_a\right)} = \exp\Big(\sum_{1<a<b<5} v'_{ab}(1 - s_a s_b)\Big)$$

durch Einsetzen der Definition der Boltzmann-Gibbs-Verteilung (5.1.9) und Kürzen der auf beiden Seiten in gleicher Weise vorhandenen von s_1 unabhängigen Wechselwirkungsterme der Knoten 2, ..., n. Die acht möglichen Kombinationen der Aktivationen $s_2, s_3, s_4 \in \{-1, 1\}$ werden durch die Symmetrie $(s_2, s_3, s_4) \leftrightarrow (-s_2, -s_3, -s_4)$ auf vier Kombinationen reduziert: eine der Aktivationen hätte also genausogut fest als Bias gewählt werden können. Die Gleichung mit $s_2 = s_3 = s_4$ ist trivial erfüllt und es verbleiben drei Gleichungen mit drei Unbekannten. Durch Logarithmieren beider Seiten werden diese Gleichungen sogar linear in den Unbekannten v'_{23}, v'_{24} und v'_{34} mit eindeutiger Lösung (6.1.3;i). Der Prozeß des Weglassens von Knoten 1 mit all seinen Gewichten unter Einbau der neuen Gewichte v'_{23}, v'_{24} und v'_{34} heiße *Sterndezimierung*, siehe Abb. 71 (die in der Physik ⟦Onsager 1944, Syozi 1972⟧ als Stern-Dreieck- oder Y-Δ-Transformation zwischen Bienenwaben- und Dreiecksspingittern vermittelt).

(ii) Auf gleiche Weise kann der Fall $k = 3$ gelöst werden; es ergibt sich eine Gleichung mit der Unbekannten v'_{23}. Die Lösung lautet

$$v'_{23} = \log\left(\frac{\cosh(v_{12} + v_{13})}{\cosh(v_{12} - v_{13})}\right)/2.$$

Dies hätte auch $v_{14} = 0$ in (6.1.3;i) ergeben mit den zusätzlichen Gewichten $v'_{34} = v'_{24} = 0$, die nach der Parallelenregel (6.1.3;iv) nicht stören. Der Fall $k = 2$ kann als Spezialfall von $k = 3$ mit $v_{13} = 0$ aufgefaßt werden; die zusätzliche Wechselwirkung zwischen Knoten 2 und 3 verschwindet hier wie im trivialen Fall $k = 1$.

(iii) Ansatz (6.1.2) führt auf ein lineares Gleichungssystem mit $2^{k-2} - 1$ Gleichungen ($k - 2$ binäre Belegungen abzüglich einer trivial erfüllten Gleichung) und $(k - 1)(k - 2)/2$ Unbekannten (die paarweisen Gewichte zwischen den Knoten 2, 3, ..., k). Für $k > 4$ ist das Gleichungssystem überbestimmt und i. allg. unlösbar, wenn die ursprünglichen Gewichte v_{12}, ..., v_{1k} keinen Restriktionen unterliegen. ∎

Korollar: Dezimierung in Reihe ⟦Saul und Jordan 1994⟧ (6.1.5) *Ist Knoten* 1 *nur mit den Knoten* 2 *und* 3 *verbunden, so kann Knoten* 1 *unter Einfügen des Gewichts* v'_{23} *mit* $\tanh(v'_{23}) = \tanh(v_{12})\tanh(v_{13})$ *dezimiert werden; siehe auch Abb. 72a.* □

Beweis (6.1.6) Beachte (6.1.4;ii) und die Definition von tanh. ■

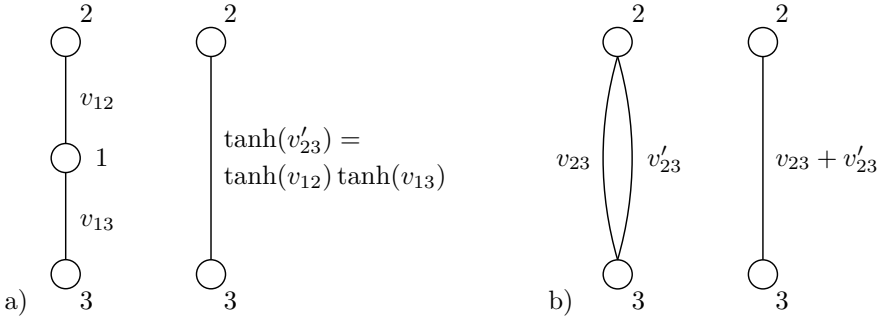
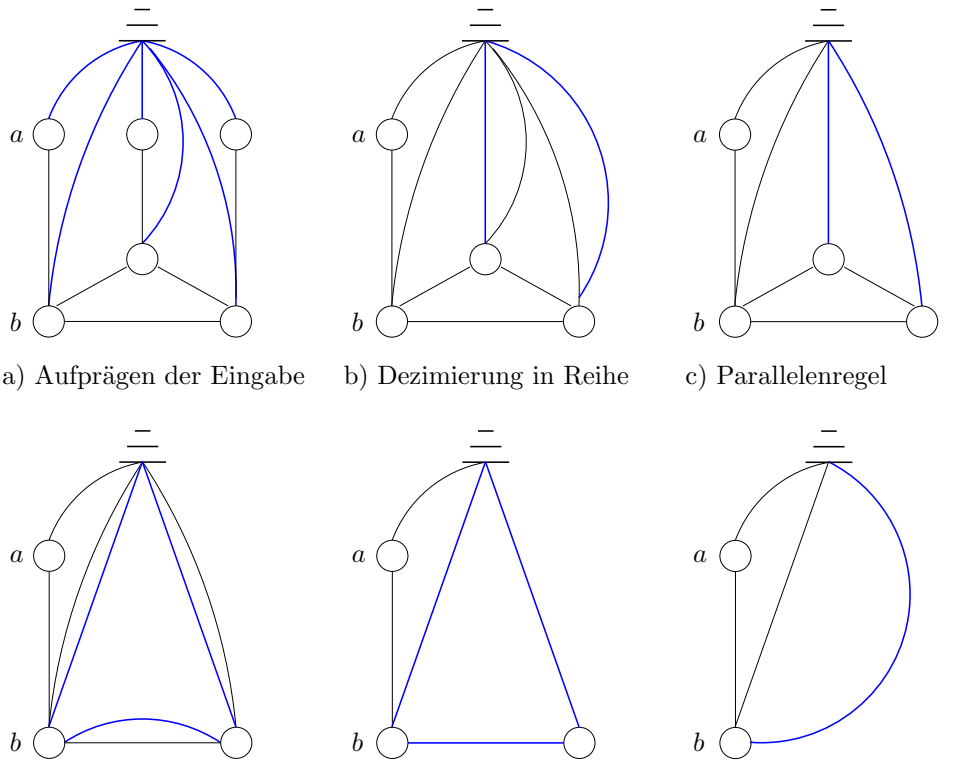


Abb. 72 Dezimierung in Reihe und Parallelenregel

Exakte Berechnung der Wechselwirkungen (6.1.7) Alle Lernregeln für die Boltzmann-Maschine haben gemein, in Termen der Wechselwirkung $\mathrm{E}(s_a s_b)_\delta$ zwischen Knoten oder der Erwartungswerte $\mathrm{E}(s_a)_\delta$ von Aktivationen formuliert zu sein. Werden in einer Boltzmann-Maschine die Knoten a, b und der Biasknoten 0 festgehalten, so kann es durch sukzessive Anwendung der Dezimierung in Reihe, der Sterndezimierung und der Parallelenregel möglich sein, das gesamte Netz auf die drei festgehaltenen Knoten zu reduzieren, siehe Abb. 73.

(i) Das Aufprägen des Musters $\delta \in \{-1, 1\}^X$ mit $X \subset N$ wird realisiert, indem die entsprechenden Knoten in X weggelassen werden, und die von diesen Knoten ausgehenden Gewichte in die Biasgewichte der Zielknoten subsummiert werden: Sei $x \in X$; das Biasgewicht w_{oy} wird um w_{xy} erhöht, wenn $s_x = 1$, und um w_{xy} erniedrigt, wenn $s_x = -1$. Das wird für alle $(x, y) \in E$ mit $x \in X$ durchgeführt.

(ii) Algorithmisch ist es sinnvoll, zuerst Knoten wegzulassen, die maximal eine Verbindung zu anderen Knoten haben. Danach wird solange die Dezimierung in Reihe angewendet, wie es Knoten mit genau zwei Verbindungen



a) Aufprägen der Eingabe b) Dezimierung in Reihe c) Parallelenregel

d) Sterndezimierung e) Parallelenregel f) Dezimierung in Reihe

Abb. 73 Dezimierung einer Boltzmann-Maschine

im verbleibenden Netz ohne die Knoten a, b und 0 gibt. Gegebenenfalls wird en passant die Parallelenregel angewendet, so daß gar nicht erst parallele Kanten eingefügt werden (Abb. 73 zeigt der Klarheit halber diesen Schritt explizit).

(iii) In einem weiteren Durchgang wird solange abwechselnd die Sterndezimierung und die Dezimierung in Reihe (jeweils incl. Parallelenregel) angewendet bis kein Knoten im verbleibenden Netz ohne a, b und 0 mehr dezimiert werden kann.

Ziel ist es, dann das bezüglich der Wechselwirkung $\mathrm{E}(s_a s_b)_\delta$ äquivalente Netz aus Abb. 74a zu bekommen. Hier berechnet sie sich einfach zu

$$\mathrm{E}(s_a s_b)_\delta = \sum_{s_a, s_b} P_{vT}^{74\mathrm{a}}(s_a, s_b) s_a s_b$$

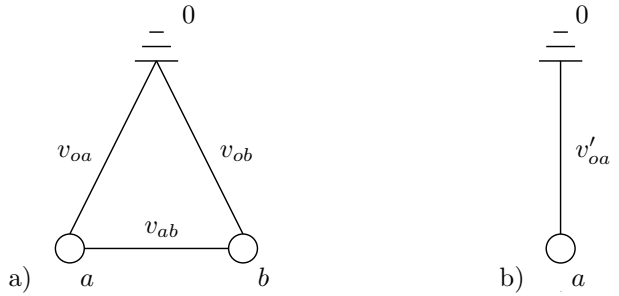


Abb. 74 Reduzierte Netze zur Berechnung von a) $\mathrm{E}(s_a s_b)$ und b) $\mathrm{E}(s_a)$

$$= \frac{\exp(v_{ab})\cosh(v_{oa}+v_{ob}) - \exp(-v_{ab})\cosh(v_{oa}-v_{ob})}{\exp(v_{ab})\cosh(v_{oa}+v_{ob}) + \exp(-v_{ab})\cosh(v_{oa}-v_{ob})}.$$

Ist nur der Erwartungswert $\mathrm{E}(s_a)_\delta$ gesucht, so muß nur noch Knoten b dezimiert werden. Mit dem Netz aus Abb. 74b erhält man

$$\mathrm{E}(s_a)_\delta = \sum_{s_a} P^{74\mathrm{b}}_{v'T}(s_a)s_a = \tanh(v'_{oa}). \qquad \square$$

Dezimierbare Boltzmann-Maschinen (6.1.8) Eine Boltzmann-Maschine, die nach dem Aufprägen einer Eingabe durch sukzessive Dezimierung in Reihe, Sterndezimierung und Parallelenregel in das triviale Netz $\{0\}$ überführt werden kann, heiße *dezimierbar*. Diejenigen dezimierbaren Boltzmann-Maschinen, die bereits mit der Dezimierungsregel (6.1.5) von Saul und Jordan auskommen, heißen *Boltzmann-Bäume*. $\square$

Bemerkungen (6.1.9) (i) Von einem operationellen Standpunkt aus ergäbe sich als natürliche Forderung an eine dezimierbare Boltzmann-Maschine, daß sie nach Aufprägen eines Eingabemusters *für jede Kante* $(a,b) \in E$ mit dem Algorithmus (6.1.7) in ein Netz mit den Knoten a, b und 0 überführt werden kann. Dies ist eine formal stärkere Forderung als sie obige Definition erhebt. Im Vorgriff auf die Techniken des folgenden Abschnitts, insbesondere auf (6.2.14), erweist sich die stärkere Forderung jedoch als nicht notwendig. (ii) Die Dezimierbarkeit einer gegebenen Boltzmann-Maschine kann einfach dadurch geprüft werden, daß versucht wird, sie mit Algorithmus (6.1.7) zu dezimieren. $\square$

6.2 Die Zustandssumme

Die Zustandssumme wurde in Abschnitt 5.1 als Normierungsfaktor eingeführt. Ihr konkreter Wert scheint unerheblich zu sein, ihre Rolle scheint sich in der Normierung der Boltzmann-Gibbs-Verteilung zu beschränken. In der statistischen Mechanik wird der *funktionalen Abhängigkeit* der Zustandssumme von ihren Parametern jedoch zu Recht eine grundlegende Bedeutung zugemessen.

In diesem Abschnitt wird die Relevanz der Abbildung $w \mapsto Z_w$ für Boltzmann-Maschinen dargestellt. Aus ihr kann nämlich die Boltzmann-Gibbs-Verteilung zurückgewonnen werden und damit das gesamte Verhalten des Systems. Insbesondere können Erwartungswerte, Momente und Korrelationsfunktionen der zugrundeliegenden Verteilung aus partiellen Ableitungen von $w \mapsto \log(Z_w)$ berechnet werden. Das ist besonders von Vorteil, da Differentiation in der Regel einfacher ist als Summation bzw. Integration.

Trotzdem muß zu Beginn einmal auch die Abbildung $w \mapsto Z_w$ algorithmisch bestimmt werden: Im allgemeinen ist das intraktabel bezüglich der Anzahl der Knoten. Für dezimierbare Boltzmann-Maschinen wird in diesem Abschnitt jedoch ein Algorithmus vorgestellt, dessen Aufwand höchstens quadratisch mit Anzahl der Knoten wächst.

Die folgenden wahrscheinlichkeitstheoretischen Ausführungen dienen der Einführung des wichtigen Begriffs der *Kumulante*. Dabei werden Basisbegriffe der Lehrbuchliteratur ⟦z. B. Krengel 1991, Pfanzagl 1991, Bauer 1990, Bauer 1991, Frieden 1991⟧ wie Zufallsvariable, Wahrscheinlichkeitsmaß, fast überall, Dichte und stetige Verteilung vorausgesetzt.

Charakteristische Funktion (6.2.1) Sei S eine $\mathbb{R}^n$-wertige Zufallsvariable mit zugehörigem Wahrscheinlichkeitsmaß P_S. Dann ist die *charakteristische Funktion* ϕ_S von S definiert als

$$\begin{aligned}\phi_S &: \mathbb{R}^n \to \mathbb{C} \\ &x \mapsto \mathrm{E}\left(\exp(ixS)\right) \\ &\quad = \text{ z. B. } \begin{cases} \int p(s)\exp(ixs)ds & \text{falls } p \text{ Dichte von } P_S, \\ \sum_s P_S(s)\exp(ixs) & \text{falls } P_S \text{ diskret ist.} \end{cases}\end{aligned}$$

Sie hat folgende Eigenschaften ⟦Bauer 1991, § 22, Feller 1971, Kap. 15⟧:

(i) $\phi_S(0) = 1$

(ii) $|\phi_S(x)| \leq 1$ für alle $x \in \mathbb{R}^n$

(iii) Existieren die *gemischten Momente* $\mathrm{E}\left(\prod_k S_k^{m_k}\right)$ für alle $m_1, \ldots, m_n \in \mathbb{N}_o$, dann existieren alle partiellen Ableitungen von ϕ_S und es gilt mit $r = m_1 + \ldots + m_n$ für das *Moment r-ter Ordnung*

$$\mathrm{E}\left(\prod_k S_k^{m_k}\right) = (-i)^r \frac{\partial^r \phi_S(x)}{\partial x_1^{m_1} \ldots \partial x_n^{m_n}} \Big|_{x=0}.$$

(iv) Sei die Verteilung P_S von S mit einer Dichte p gegeben. Dann ist die *Fourierinverse*

$$s \mapsto \frac{1}{(2\pi)^n} \int \phi_S(x) \exp(-ixs) dx$$

von ϕ_S fast überall gleich p.

(v) Sind die Komponenten der Zufallsvariable S unabhängig voneinander und haben stetige Verteilungen mit Dichte p_i, so faktorisiert die Dichte $p(s_1, \ldots, s_n) = p_1(s_1) \ldots p_n(s_n)$ und für die charakteristischen Funktionen gilt entsprechendes:

$$\phi_S(x_1, \ldots, x_n) = \phi_{S_1}(x_1) \ldots \phi_{S_n}(x_n)$$

(vi) Faltung: Hat die Summe Y von unabhängigen Zufallsvariablen $S_1, \ldots, S_n$ mit charakteristischen Funktionen $\phi_{S_1}, \ldots, \phi_{S_n}$ die charakteristische Funktion ϕ_Y, dann ist

$$\phi_Y(x) = \phi_{S_1}(x) \ldots \phi_{S_n}(x).$$

(vii) Affine Transformation: Sei A eine reellwertige $n \times n$-Matrix und $b \in \mathbb{R}^n$, dann ist

$$\phi_{AS+b}(x) = \exp(ixb)\phi_S(A^T x).$$

Die Transponierung der Matrix A kommt von einer Eigenschaft des Skalarprodukts xs: als Matrixprodukt lautet es nämlich explizit $x^T \cdot s$, weswegen $x(A \cdot s) = x^T \cdot A \cdot s = (A^T \cdot x)^T \cdot s = (A^T \cdot x)s$. □

Bemerkungen (6.2.2) (i) Diese Eigenschaften zeigen, daß die charakteristische Funktion ihren Namen zu Recht erhält: Sowohl alle Momente, als auch die Dichte einer stetigen Verteilung lassen sich direkt aus ihr bestimmen; besonders wertvoll ist sie zur Bestimmung von Faltungsprodukten, die in normale Produkte übergeführt werden.

(ii) Im allgemeinen ergeben sich die Momente leichter durch Differentiation der charakteristischen Funktion als durch Integration von etwa $s^k p(s)$.

(iii) Die Abbildung $P_S \mapsto \phi_S$ ist in der Literatur auch als *Fouriertransformation von Wahrscheinlichkeitsmaßen* oder als *Fourier-Stieltjes-Transformation* bekannt. Sie ist injektiv; daher können nicht nur stetige Wahrscheinlichkeitsmaße P_S durch die Angabe von ϕ_S charakterisiert werden.

(iv) In der physikalischen Literatur sind gelegentlich die Rollen von Fouriertransformierter und Fourierinverser vertauscht. □

Kumulanten und Korrelationsfunktionen (6.2.3) Im Komplexen ist der Logarithmus mehrdeutig. Jedoch zeigt es sich ⟦Bauer 1991, § 26⟧, daß es zu einer charakteristischen Funktion ϕ_S einer $\mathbb{R}^n$-wertigen Zufallsvariable S genau eine stetige Abbildung $\Phi_S\colon \mathbb{R}^n \to \mathbb{C}$ mit $\Phi_S(0) = 0$ gibt, so daß

$$\phi_S = \exp \circ \Phi_S.$$

In diesem Sinn wird der Logarithmus von ϕ_S durch $\Phi_S := \log \circ \phi_S$ definiert. Φ_S heißt *kumulantengenerierende Funktion*. Existieren alle gemischten Momente von S, so sind sowohl ϕ_S als auch Φ_S in eine Potenzreihe um 0 entwickelbar und es ist mit $m = (m_1, \ldots, m_n)$

$$\Phi_S(x) = \sum_{r=1}^{\infty} i^r \sum_{\substack{m_1,\ldots,m_n \\ \sum m_k = r}} \mathrm{C}_m^r \prod_{k=1}^{n} \frac{x_k^{m_k}}{m_k!} \quad \text{mit}$$

$$\mathrm{C}_m^r = (-i)^r \frac{\partial^r \Phi_S(x)}{\partial x_1^{m_1} \ldots \partial x_n^{m_n}} \Big|_{x=0}.$$

Jeder Vorfaktor C_m^r heißt *Kumulante* von S der Ordnung $r = \sum m_k$ und ist eine Funktion der Momente von S. Dies ergibt sich durch Expansion des Logarithmus in eine Potenzreihe und Vergleich der Koeffizienten. Daher wird auch die symbolische Schreibweise $\mathrm{C}(S_1^{m_1} \ldots S_n^{m_n})$ statt C_m^r benutzt.

Eine Kumulante, bei der mehr als nur eine Komponente von m verschieden von Null ist (bei der mehr als eine Komponente von S involviert ist), heißt auch *Korrelationsfunktion.* □

Mit Hilfe der Korrelationsfunktionen lassen sich, wie folgender Satz zeigt, Korrelationen höherer Ordnung aufspüren.

Satz: Kumulanten und Unabhängigkeit (6.2.4) *Sei S eine $\mathbb{R}^n$-wertige Zufallsvariable mit stetiger Verteilung P_S und existieren alle gemischten Momente von S. Dann gilt:*

(i) C^1_m *sind die Erwartungswerte der Komponenten von S:* $\mathrm{C}(S_k) = \mathrm{E}(S_k)$*; die Kumulanten* C^2_m *zweiter Ordnung sind die Elemente der Varianz-Kovarianzmatrix von S:* $\mathrm{C}(S_k S_l) = \mathrm{E}(S_k S_l) - \mathrm{E}(S_k)\,\mathrm{E}(S_l)$

(ii) Sind die $S_1, \ldots, S_n$ unabhängig, so verschwinden die Korrelationsfunktionen. □

Beweis (6.2.5) (i) kann durch direktes Nachrechnen gezeigt werden:

$$\begin{aligned}\mathrm{C}(S_k) &= -i\frac{\partial \log \mathrm{E}(\exp(ixS))}{\partial x_k}\Big|_{x=0} \\ &= -i\frac{\mathrm{E}(iS_k \exp(ixS))}{\mathrm{E}((\exp(ixS))}\Big|_{x=0} = \mathrm{E}(S_k)\end{aligned}$$

Analog (durch eine weitere Differentiation) ergibt sich die Aussage über die Varianz-Kovarianzmatrixelemente. (ii) Sind die $S_1, \ldots, S_n$ unabhängig, so faktorisiert nach (6.2.1;v) die charakteristische Funktion, und Φ_S zerfällt in eine Summe

$$\Phi_S(x) = \sum_{k=1}^{m} \log \phi_{S_k}(x_k).$$

Daher verschwinden die Korrelationsfunktionen; sie enthalten nämlich Ableitungen nach zwei verschiedenen Komponenten von x. ∎

Bemerkung: Bedeutung der Kumulanten (6.2.6) Da Wahrscheinlichkeitsmaße normiert sind, müssen Dichten auf $\mathbb{R}$ für $s \mapsto \pm\infty$ verschwinden. Momente immer höherer Ordnung sagen also etwas aus über Bereiche mit immer kleineren Wahrscheinlichkeiten. Darüberhinaus gilt $\mathrm{E}(S^{2k}) \geq \mathrm{E}(S^k)^2$, so daß die geraden Momente oftmals beliebig hohe

Werte annehmen, ohne eine direkt verwertbare Interpretation zu haben. Das mag die Ursache dafür sein, daß vor allem die ersten beiden Momente – und davon abgeleitet die Varianz – in der Praxis häufig auftauchen. Die aus den Momenten gebildeten Korrelationsfunktionen jedoch haben ihre Aussage: Sie verschwinden bei unabhängigen Zufallsvariablen und geben ansonsten Auskunft über Größe und Ordnung der Korrelation (d. i. die kleinste natürliche Zahl r, für die die Korrelationsfunktionen C_m^{r+1} nicht verschwinden). Dies stellt den vielleicht der größten Beitrag der Definition von Φ_S und der Kumulanten dar. □

Der Zusammenhang obiger etablierter Mathematik zu Boltzmann-Maschinen wird durch folgenden Satz hergestellt.

Satz: Zustandssumme und Kumulanten (6.2.7) *Sei $N = \{0, 1, \ldots, n\}$ eine Boltzmann-Maschine mit Biasknoten 0 und enthalte die Kantenmenge E alle Kanten der Form $(0, a)$ mit $a > 0$. Die Aktivationen s_i induzieren eine $\{-1, 1\}^n$-wertige Zufallsvariable $S = (S_1, \ldots, S_n)$, die der Boltzmann-Gibbs-Verteilung mit Zustandssumme*

$$Z_w = \sum_{s \in \{-1,1\}^n} \exp\Big(\sum_{(a,b) \in E} w_{ab} s_a s_b / 2T \Big)$$

folgt (mit $s_o := 1$). Seien $m_1, \ldots, m_n \in \mathbb{N}_o$ und sei $r = \sum_k m_k$. Dann ist für $r > 0$ die Kumulante C_m^r von S gegeben durch

$$\mathrm{C}_m^r = (2T)^r \, \frac{\partial^r (w \mapsto \log(Z_w))}{\partial w_{01}^{m_1} \ldots \partial w_{0n}^{m_n}}. \qquad \square$$

Beweis (6.2.8) Der Gewichtungsvektor w werde aufgeteilt in einen Teilvektor $w^o := (w_{01}, \ldots, w_{0n})$ und einen anderen Teilvektor w^r, der die restlichen Komponenten enthält. Dann ist die kumulantengenerierende Funktion

$$\begin{aligned} \Phi_S(x) &= \log \left(\sum_s \frac{\exp\big(\sum w_{ab} s_a s_b / 2T \big)}{Z_w} \exp(ixs) \right) \\ &= \log \left(\sum_s \exp\Big(\big((w^o + i2T\, x)s + \textstyle\sum w^r_{ab} s_a s_b \big) / 2T \Big) \right) - \log(Z_w). \end{aligned}$$

Wegen $r > 0$ muß bei der Berechnung der Kumulante C_m^r immer nach mindestens einer Komponenten von x partiell abgeleitet werden. Die Ableitung des Subtrahenden verschwindet aber, so daß nur der Minuend beachtet werden muß. Dieser wiederum kann aufgefaßt werden als Abbildung $x \mapsto \log(Z_{w^o+i2T\,x,w^r})$. Somit ist klar, daß

$$\begin{aligned}\mathrm{C}_m^r &= (-i)^r \frac{\partial^r (x \mapsto \log(Z_{w^o+i2T\,x,w^r}))}{\partial x_1^{m_1} \ldots \partial x_n^{m_n}} \Big|_{x=0} \\ &= (2T)^r \frac{\partial^r (w^o \mapsto \log(Z_{w^o,w^r}))}{\partial w_{01}^{m_1} \ldots \partial w_{0n}^{m_n}}.\end{aligned}$$

∎

Korollar des Beweises (6.2.9) *Die charakteristische Funktion einer nach der Boltzmann-Gibbs-Verteilung mit Zustandssumme Z_w verteilten Zufallsvariable ist die Abbildung $x \mapsto Z_{w^o+i2Tx,w^r}/Z_w$ von $\mathbb{R}^n \to \mathbb{C}$.* □

Ein Algorithmus zur Berechnung von $w \mapsto Z_w$ kann also auch zur Berechnung der charakteristischen Funktion eingesetzt werden. Folgendes Lemma ist die Grundlage zu solch einem Algorithmus.

Lemma: Änderung der Zustandssumme (6.2.10) *$1, \ldots, k, \ldots, n$ ($1 \le k \le 4$) seien die Knoten einer Boltzmann-Maschine $(N, E, w, T \neq 0)$. Knoten 1 sei bipolar und nur mit den $k-1$ Knoten 2, …, k verbunden. Die Knoten 2, …, n seinen bipolar bis auf einen, den Biasknoten. Die Zustandssumme der Boltzmann-Maschine sei Z_w^{a}, die der gemäß Satz (6.1.3) um den Knoten 1 dezimierten Boltzmann-Maschine sei $Z_{w'}^{\mathrm{b}}$. Dann gilt mit $v = w/T$*

$$\frac{Z_w^{\mathrm{a}}}{Z_{w'}^{\mathrm{b}}} = \begin{cases} 2\left(\prod_{s_3,s_4} \cosh(v_{12} + v_{13}s_3 + v_{14}s_4)\right)^{1/4} & \text{falls } k = 4 \\ 2\left(\cosh(v_{12}+v_{13})\cosh(v_{12}-v_{13})\right)^{1/2} & \text{falls } k = 3 \\ 2\cosh(v_{12}) & \text{falls } k = 2 \\ 2 & \text{falls } k = 1. \end{cases}$$

□

Beweis (6.2.11) Für $k = 4$ sind die Voraussetzungen des Satzes (6.1.3) gegeben. Der dann erfüllbare Ansatz der Dezimierung (6.1.2) ergibt

$$\frac{1}{Z^{\mathrm{a}}_w} \sum_{s_1} \exp\left(s_1 \sum_{a=2}^{4} v_{1a} s_a \right) = \frac{1}{Z^{\mathrm{b}}_{w'}} \exp\left(\sum_{1<a<b<5} v'_{ab} s_a s_b \right).$$

Das Einsetzen der Lösung (6.1.3;i) für v'_{ab} resultiert im ersten Fall von (6.2.10). Da die Zustandssumme unabhängig vom Zustand s sein muß, reicht es, im obigen Gleichungssystem die Gleichung für $s_2 = s_3 = s_4 = 1$ zu betrachten. Die Fälle $k = 3$ und $k = 2$ sind wieder Spezialfälle von $k = 4$ mit $v_{14} = 0$ bzw. $v_{13} = v_{14} = 0$. Der Fall $k = 1$ behandelt die Entfernung eines isolierten Knotens: hier ist aber unmittelbar klar, daß $Z^{\mathrm{a}}_w = 2Z^{\mathrm{b}}_{w'}$. ∎

Zustandssumme dezimierbarer Boltzmann-Maschinen (6.2.12) *Die Zustandssumme Z^{a}_w einer dezimierbaren Boltzmann-Maschine kann dadurch exakt berechnet werden, daß die gemäß Lemma (6.2.10) durch das Dezimieren entstehenden Terme $Z^{\mathrm{a}}_w/Z^{\mathrm{b}}_{w'}$, $Z^{\mathrm{b}}_{w'}/Z^{\mathrm{c}}_{w''}$, …, miteinander multipliziert werden bis die triviale Boltzmann-Maschine mit Zustandssumme* 1 *erreicht ist.* □

Aufwand der Berechnung (6.2.13) Die Berechnung der Zustandssumme hat einen in der Anzahl der Knoten höchstens quadratischen Aufwand: Im schlimmsten Fall müssen nämlich alle Knoten auf ihre Dezimierbarkeit hin untersucht werden, bis einer entfernt werden kann. Die Dezimierung eines Knotens selbst beschränkt sich auf einige Operationen, so daß letztendlich zur vollständigen Dezimierung höchstens $|N|(|N|-1)/2$ Zeitintervalle benötigt werden. □

Partielle Ableitung der Zustandssumme (6.2.14) Die partiellen Ableitungen der Zustandssumme können nach Satz (6.2.7) gewinnbringend ausgenutzt werden: z. B. ist $\mathrm{E}(s_a) = 2\,\partial(v \mapsto \log(Z_{vT}))/\partial v_{oa}$ und $\mathrm{E}(s_a s_b) = 2\,\partial(v \mapsto \log(Z_{vT}))/\partial v_{ab}$. Die v-Abhängigkeit der Zustandssumme ist jedoch nur implizit durch den induktiven Algorithmus (6.2.12) gegeben.

Um die partiellen Ableitungen zu berechnen bieten sich zwei Wege an: Sie können numerisch als Differenzenquotient genähert werden; oder aber es

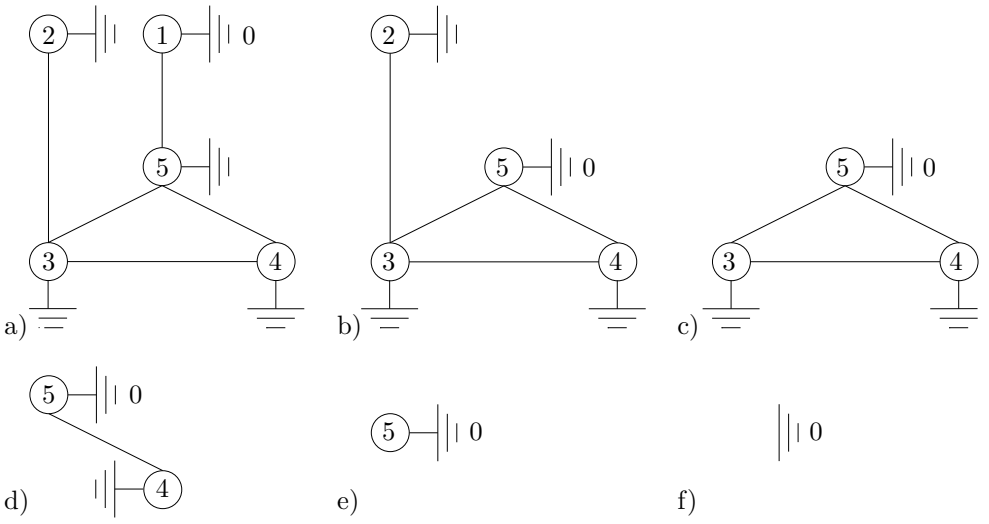


Abb. 75 Restlose Dezimierung einer Boltzmann-Maschine

wird der Dezimierungsvorgang in (6.2.12) als Abbildungsnetz mit der Netzabbildung $v \mapsto \log(Z_{vT})$ formuliert. Dies ist in Abb. 76 für die Dezimierung des Netzes in Abb. 75a veranschaulicht. Danach kann die Kettenregel in Abbildungsnetzen (1.1.8) zur Bildung der partiellen Ableitung ausgenutzt werden.

Letzterer Weg ist genauer und schneller: Es muß das Abbildungsnetz von Abb. 76 ja nur einmal vorwärts und einmal rückwärts durchlaufen werden, um *alle* $|E|$ partiellen Ableitungen gleichzeitig bestimmen zu können. Für die numerische Variante wären $|E| + 1$ Berechnungen der Zustandssumme nötig. Für die Berechnung aller Wechselwirkungen $\mathrm{E}(s_a s_b)$ gemäß (6.1.7) müßten übrigens auch $|E|$ Dezimierungsvorgänge durchgeführt werden. Der Weg über das Abbildungsnetz ist also jedenfalls empfehlenswert. □

6.3 Effizientes Schließen

Im folgenden werden einige Techniken zur Berechnung der zum Schließen relevanten Größe

$$p_w(\gamma|\alpha) = \sum_{\beta} P_w(\beta\gamma|\alpha)$$

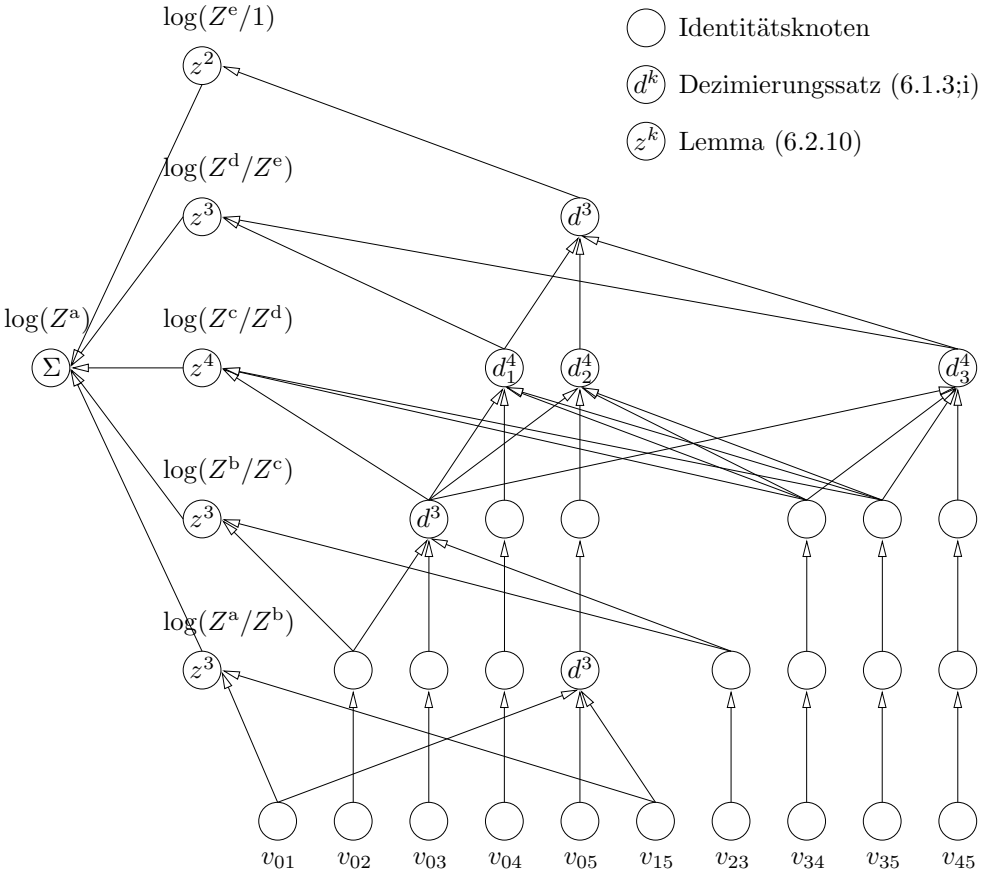


Abb. 76 Abbildungsnetz für die Berechnung der Zustandssumme

gegeben, von denen Abb. 77 zeigt, wann sie am besten angewendet werden können. Dabei wird angenommen, daß die zugrundeliegende Boltzmann-Maschine dezimierbar ist. Die zweifache Dezimierung (6.3.4) kann immer angewendet werden, in manchen Fällen ist aber eine der anderen Techniken leicht im algorithmischen Vorteil.

Alle Techniken haben das gemeinsame Element, daß bedingte Wahrscheinlichkeiten durch das Aufprägen des Musters realisiert werden: Sei O die Menge der Ausgabe-, U die der Binnen- und I die der Eingabeknoten. Durch Aufprägen eines Musters α an die Eingabeknoten reduziert sich

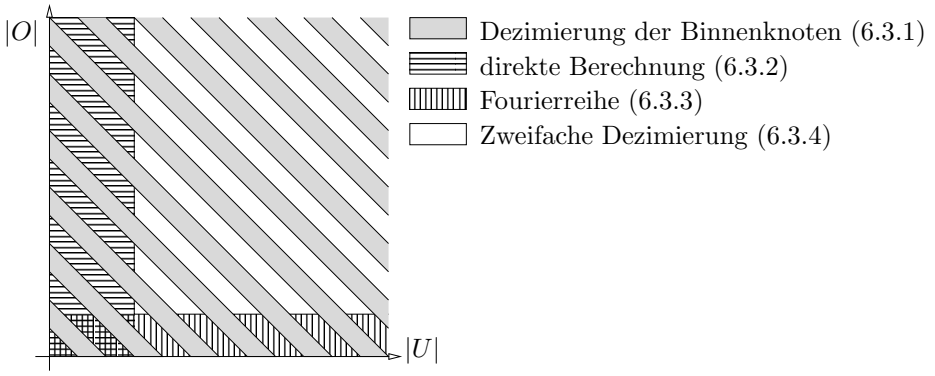


Abb. 77 Verschiedene Techniken zur Berechnung von $p_w(\gamma|\alpha)$ in Abhängigkeit von der Anzahl $|U|$ der Binnenknoten und der Anzahl $|O|$ der Ausgabeknoten

die Boltzmann-Maschine[♮] $N = I \cup U \cup O \cup \{0\}$ zur Boltzmann-Maschine $N^\alpha := U \cup O \cup \{0\}$, deren Boltzmann-Gibbs-Verteilung $P_w(\beta\gamma|\alpha)$ ist. Alle Größen der Boltzmann-Maschine N^α erhalten einen Hochindex α, um sie einerseits von entsprechenden Größen der Boltzmann-Maschine N zu unterscheiden und um andererseits explizit auf die Abhängigkeit des aufgeprägten Musters α hinzuweisen.

Dezimierung der Binnenknoten (6.3.1) Eine sehr effiziente Möglichkeit der Berechnung von $p_w(\gamma|\alpha)$ ergibt sich, wenn bei der Boltzmann-Maschine N^α zunächst alle Binnenknoten dezimiert werden können, so daß die entstehende Boltzmann-Maschine $\tilde{N}^\alpha = O \cup \{0\}$ dezimierbar ist. Eine Boltzmann-Maschine, bei der das möglich ist, heiße *binnendezimierbar*. Die Menge der binnendezimierbaren Boltzmann-Maschinen ist eine echte Teilmenge der dezimierbaren: Die Netze D_1, D_2 und D_5 aus Abb. 78 sind dezimierbar, nicht aber binnendezimierbar.

Sei N^α binnendezimierbar. Das Dezimieren der Binnenschicht von N^α mit resultierender Boltzmann-Maschine $\tilde{N}^\alpha$ bewirkt nun genau das gewünschte Bilden der Randwahrscheinlichkeit von $P_w(\beta\gamma|\alpha)$ über $\beta \in \{-1, 1\}^U$: Die

[♮] zur Vereinfachung wird N statt (N, E, w, T) geschrieben

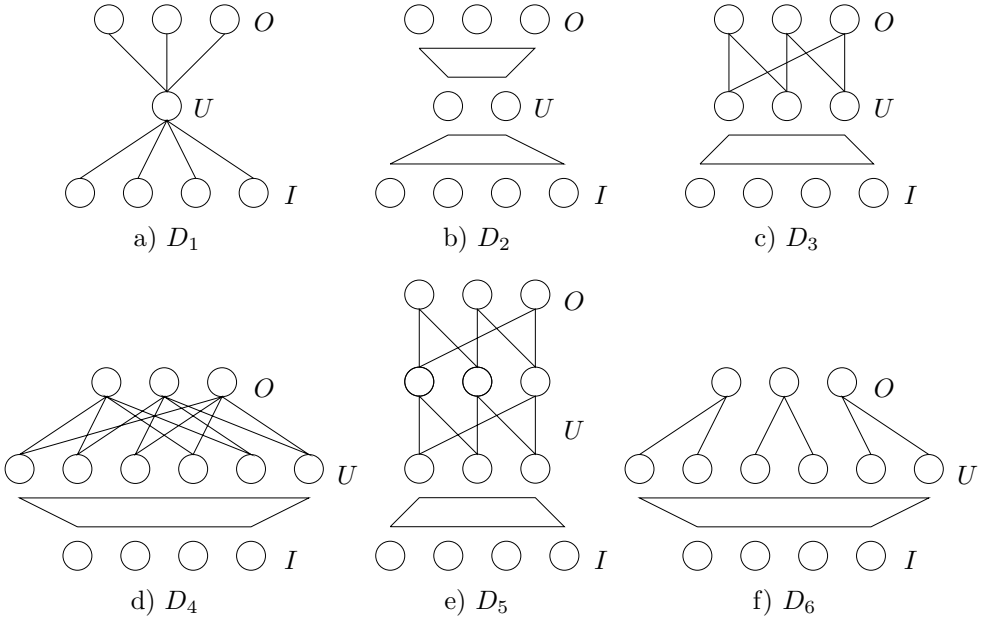


Abb. 78 Einige dezimierbare Boltzmann-Maschinen (weder der Biasknoten noch die Biaskanten der Knoten $U \cup O$ sind eingezeichnet; das Trapezsymbol bedeute vollständige Verknüpfung)

Boltzmann-Gibbs-Verteilung

$$p_w(\gamma|\alpha) = \sum_\beta P_w(\beta\gamma|\alpha)$$
$$= \tilde{p}^\alpha_{\tilde{w}^\alpha}(\gamma) = \frac{\exp(-\tilde{H}^\alpha(\tilde{w}^\alpha, \gamma)/T)}{\tilde{Z}^\alpha_{\tilde{w}^\alpha}}$$

von $\tilde{N}^\alpha$ läßt sich nun effizient berechnen, zumal ja die Zustandssumme $\tilde{Z}^\alpha_{\tilde{w}^\alpha}$ der dezimierbaren Boltzmann-Maschine $\tilde{N}^\alpha$ leicht auszuführen ist. □

Direkte Berechnung (6.3.2) Ist die Anzahl $|U|$ der Binnenknoten klein, so kann

$$p_w(\gamma|\alpha) = \sum_\beta P^\alpha_{w^\alpha}(\beta\gamma) = \frac{1}{Z^\alpha_{w^\alpha}} \sum_\beta \exp(-H^\alpha(w^\alpha, \beta\gamma)/T)$$

durch direktes Summieren über $\beta \in \{-1,1\}^U$ berechnet werden. Die Zustandssumme $Z^\alpha_{w^\alpha}$ bereitet kein Problem, vorausgesetzt die zugrundeliegende Boltzmann-Maschine ist dezimierbar. □

Fourierreihe (6.3.3) Die sofort aus der Definition folgende Transformationseigenschaft (6.2.1;vii) der charakteristischen Funktion für eine affine Abbildung der Zufallsvariable kann im Fall weniger Ausgabeknoten ausgenutzt werden: Sei BΓ die durch die Aktivation $\beta\gamma$ in N^α induzierte Zufallsvariable. Betrachte die Projektion $\Pi_O\colon \mathbb{R}^{U\cup O} \to \mathbb{R}^O$ von $\beta\gamma$ auf γ. Sei $\phi^\alpha_{\mathrm{B}\Gamma}$ die nach Korollar (6.2.9) effizient zu berechnende charakteristische Funktion der Zufallsvariable BΓ in der dezimierbaren Boltzmann-Maschine N^α. Dann ist die charakteristische Funktion ϕ^α_Γ von $\Gamma = \Pi_O \circ \mathrm{B}\Gamma$ gemäß (6.2.1;vii) einfach durch

$$\phi^\alpha_\Gamma = \phi^\alpha_{\mathrm{B}\Gamma} \circ \Pi_O^T$$

gegeben, wobei Π_O^T die Transponierte der linearen Abbildung Π_O ist. Durch Rücktransformation von ϕ^α_Γ erhält man die gesuchten Wahrscheinlichkeiten $p_w(\gamma|\alpha)$. Abb. 79 zeigt diese Idee als kommutierendes Diagramm.

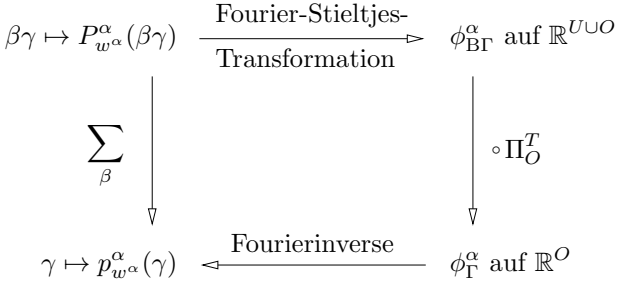


Abb. 79 Kommutierendes Diagramm für die Randverteilung

Die Rücktransformation der charakteristischen Funktion ϕ^α_Γ einer diskreten auf $\mathbb{Z}^O$ konzentrierten Zufallsvariable erfolgt i. allg. durch die Fourierinverse

$$\gamma \mapsto \frac{1}{(2\pi)^{|O|}} \int_{[-\pi,\pi]^O} \exp(-i\gamma x)\phi^\alpha_\Gamma(x)dx.$$

In diesem speziellen Fall ist γ aber auf das endliche Gitter $\{-1,1\}^O$ beschränkt, und die Fourierreihe

$$p_w(\gamma|\alpha) = \frac{1}{2^{|O|}} \sum_{x \in \{0,\pi/2\}^O} \exp(-i\gamma x)\phi_\Gamma^\alpha(x)$$

genügt zur exakten Berechnung der gesuchten Wahrscheinlichkeiten. Die Summation kann bei wenigen Ausgabeknoten aber direkt♮ ausgeführt werden. Die Gültigkeit dieser Gleichung kann durch Einsetzen der Definition von $\phi_\Gamma^\alpha = \sum_{\gamma'} \exp(ix\gamma')p_w(\gamma'|\alpha)$ und der Beobachtung von

$$\frac{1}{2^{|O|}} \sum_{x \in \{0,\pi/2\}^O} \exp(i(\gamma' - \gamma)x) = \delta_{\gamma'\gamma}$$

sofort eingesehen werden. □

Zweifache Dezimierung (6.3.4) Bei dezimierbaren Boltzmann-Maschinen kann immer wie folgt ein traktabler Ausdruck erzielt werden: Durch Einsetzen der Definition für bedingte Wahrscheinlichkeiten ergibt sich, daß für jedes $\beta \in \{-1,1\}^U$ gilt

$$\begin{aligned} p_w(\gamma|\alpha) &= \frac{P_w(\beta\gamma|\alpha)}{P_w(\beta|\alpha\gamma)} = \frac{P_{w^\alpha}^\alpha(\beta\gamma)}{P_{w^{\alpha\gamma}}^{\alpha\gamma}(\beta)} \\ &= \frac{\exp(-H^\alpha(w^\alpha, \beta\gamma)/T)}{\exp(-H^{\alpha\gamma}(w^{\alpha\gamma}, \beta)/T)} \cdot \frac{Z_{w^{\alpha\gamma}}^{\alpha\gamma}}{Z_{w^\alpha}^\alpha} \\ &= \exp\Big(\sum_{c \in O} \gamma_c w_{oc}/T + \sum_{\substack{(a,c)\in E \\ a\in I, c\in O}} w_{ac}\alpha_a\gamma_c/T\Big) \frac{Z_{w^{\alpha\gamma}}^{\alpha\gamma}}{Z_{w^\alpha}^\alpha}. \end{aligned}$$

Anhand von Abb. 80 kann nachvollzogen werden, warum der Energieunterschied $H^{\alpha\gamma}(w^{\alpha\gamma}, \beta) - H^\alpha(w^\alpha, \beta\gamma)$ von β unabhängig ist und bis auf Temperaturskalierung durch obiges Argument der Exponentialfunktion gegeben ist. Es wird bei einer Boltzmann-Maschine N mit einem Eingabe-, einem Binnen- und einem Ausgabeknoten (Abb. 80a) zuerst ein Eingabemuster α aufgeprägt mit resultierender Boltzmann-Maschine N^α (Abb. 80b) und danach sowohl ein Eingabe-, als auch ein Ausgabemuster (Abb. 80c). Die

♮ im Fall der Boltzmann-Maschinen nützt die Technik der Fast-Fourier-Transformation nichts, weil eine Aktivation nur zwei Werte annehmen kann

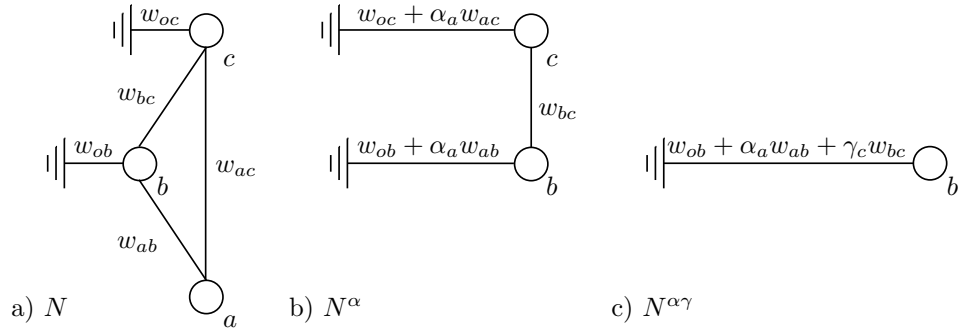


Abb. 80 Die Boltzmann-Maschinen N, N^α und $N^{\alpha\gamma}$ an einem Beispiel

beteiligten Energien lassen sich anhand der Gewichtungsvektoren w^α und $w^{\alpha\gamma}$ ablesen. □

6.4 Relevanz dezimierbarer Boltzmann-Maschinen

Hidden-Markow-Modelle werden oft als statistische Modelle diskreter Zeitreihen eingesetzt. Ein Grund dafür mögen die effizienten Lernregeln für Hidden-Markow-Modelle sein ⟦Baum 1972⟧. Auch im Bereich der neuronalen Netze werden sie benutzt, z. B. bei der Spracherkennung ⟦Juang und Rabiner 1991⟧. In diesem Abschnitt wird, der Argumentation von ⟦Saul und Jordan 1995, MacKay 1996⟧ folgend, gezeigt, daß mehrwertige Boltzmann-Maschine die Hidden-Markow-Modelle als Spezialfall enthalten.

Mehrwertige Boltzmann-Maschinen (6.4.1) Die Aktivation s_a des Knotens a einer Boltzmann-Maschine kann zwei Werte haben, -1 und 1. Bei *mehrwertigen Boltzmann-Maschinen* kann jeder Knoten i außer dem Biasknoten individuell viele, $m_a > 1$ Werte $1, \ldots, m_a$ annehmen.

Das zwischen zwei Knoten a und b vermittelnde Gewicht ist dann eine $m_a \times m_b$-Matrix w_{ab}, deren Komponente $w_{ab}(s_a, s_b)$ den negativen Energieanteil der Kante (a, b) beschreibt, wenn Knoten a die Aktivation s_a und Knoten b die Aktivation s_b zeigt. Die Forderung nach Symmetrie der Gewichte bedeutet dann $w_{ab} = w_{ba}^T$, die Gewichtsmatrix w_{ab} selbst ist beliebig, und der Gewichtungsvektor w besteht aus Gewichtsmatrizen. Die Energie

einer mehrwertigen Boltzmann-Maschine mit Gewichtungsvektor w und Zustand s ist i. w. durch die Summe aller Energieanteile gegeben:

$$H(w,s) := -\frac{1}{2} \sum_{(a,b)\in E} w_{ab}(s_a, s_b) \qquad \square$$

Bemerkung (6.4.2) *Zweiwertige Boltzmann-Maschinen* sind nun mehrwertige Boltzmann-Maschinen, deren Knoten (außer dem Biasknoten) immer zwei Werte annehmen können. Die ursprüngliche Boltzmann-Maschine (5.1.1) ist ein Spezialfall von zweiwertigen Boltzmann-Maschinen, nämlich mit Gewichtsmatrizen der inneren Struktur

$$w_{ab} = \begin{pmatrix} u_{ab} & -u_{ab} \\ -u_{ab} & u_{ab} \end{pmatrix} \quad \text{und} \quad w_{oa} = (-u_{oa}, u_{oa})\,.$$

Das spiegelt genau die negativen Energieanteile $u_{ab}s_as_b$ bzw. $u_{oa}s_a$ der ursprünglichen Boltzmann-Maschine mit Gewichtungsvektor u wider. Zu beachten ist jedoch, daß selbst die zweiwertige Boltzmann-Maschine mehr Variablen besitzt als die ursprüngliche Boltzmann-Maschine, denn die Gewichtsmatrixelemente können ja unabhängig belegt werden. Die Dezimierung mehrwertiger Boltzmann-Maschinen muß also etwas anders sein als die der ursprünglichen. $\square$

Dezimierung mehrwertiger Boltzmann-Maschinen (6.4.3) *Sei eine mehrwertige Boltzmann-Maschine mit dem effektiven Gewichtungssatz $v = w/T$ gegeben.*

(i) Ein Knoten c, der nur mit dem Biasknoten 0 und dem Knoten a verbunden ist, kann dezimiert und durch eine effektive Gewichtsmatrix v'_{oa} ersetzt werden (siehe Abb. 81a), für die gilt

$$v'_{oa}(1, s_a) = \log\left(\sum_{s_c=1}^{m_c} \exp(v_{oc}(1, s_c) + v_{ac}(s_a, s_c))\right).$$

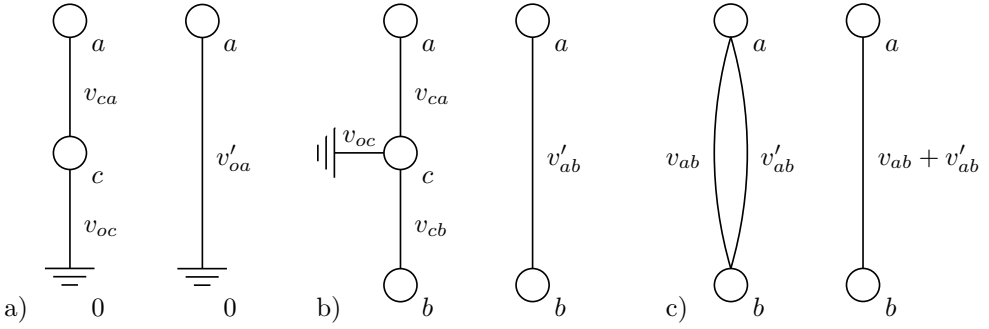


Abb. 81 Dezimierung mehrwertiger Boltzmann-Maschinen

(ii) Ein Knoten c, der nur mit dem Biasknoten 0 und zwei anderen Knoten a und b verbunden ist, kann dezimiert und durch eine effektive Gewichtsmatrix v'_{ab} ersetzt werden (siehe Abb. 81b), für die gilt

$$v'_{ab}(s_a, s_b) = \log\left(\sum_{s_c=1}^{m_c} \exp(v_{oc}(1, s_c) + v_{ac}(s_a, s_c) + v_{bc}(s_b, s_c))\right).$$

(iii) Parallelenregel: Eine neue Gewichtsmatrix v'_{ab} aus (i) muß gegebenenfalls zu einer bereits bestehenden Gewichtsmatrix v_{ab} und dessen Transponierte zur Gewichtsmatrix v_{ba} addiert werden (siehe Abb. 81c). Gab es keine Kante (a, b), so sind Kanten (a, b) und (b, a) mit jeweils der Gewichtsmatrix v'_{ab} bzw. ihrer Transponierten einzufügen. □

Beweis (6.4.4) (iii) folgt sofort aus der Linearität der Energie (6.4.1) in mehrwertigen Boltzmann-Maschinen und sowohl (i) als auch (ii) aus Ansatz (6.1.2) mit der erfüllbaren Bedingung, daß die Zustandssumme sich nicht verändert. ∎

Bemerkung: keine weiteren Dezimierungsregeln (6.4.5) Die Sterndezimierung läßt sich nicht auf mehrwertige Boltzmann-Maschinen übertragen, wie im folgenden ausgeführt wird. Das Gleichungssystem

$$\sum_{2 \le a < b < k} v'_{ab}(s_a, s_b) + \sum_{a=2}^{k-1} v'_{oa}(1, s_a) + \log\left(\frac{Z^{\mathrm{a}}_w}{Z^{\mathrm{b}}_{w'}}\right) =$$

$$\log\left(\sum_{s_1=1}^{m_1} \exp(v_{01}(1, s_1) + v_{12}(s_1, s_2) + \ldots + v_{1\,k-1}(s_1, s_{k-1}))\right),$$

das aus Ansatz (6.1.2) zur Dezimierung eines Knotens mit $k-1$ Nachbarknoten 0 (Biasknoten), 2, ..., $k-1$ entsteht, hat zunächst $m_2 m_3 \ldots m_{k-1}$ Gleichungen durch Einsetzen aller möglichen Werte für s_2, ..., s_{k-1}. Die Zahl der Unbekannten ist höchstens

$$\sum_{2 \le i < j < k} m_i m_j + \sum_{i=2}^{k-1} m_i,$$

denn mehr Gewichte können nicht zwischen den Knoten 2, ..., $k-1$ und 0 vermitteln (zumindest nicht beim Ansatz paarweiser Wechselwirkungen). Zu viele Unbekannte schaden nicht, denn Gewichtsmatrixelemente der neuen Gewichte können zu Null gesetzt werden. Aber durch zu viele Gleichungen wird das System schnell überbestimmt und damit unlösbar.

Selbst für $m_2 = \ldots = m_{k-1} = 2$, also für zweiwertige Boltzmann-Maschinen, $k = 4$ und dem Fall der invarianten Zustandssumme, wo zunächst die Zahl der Gleichungen kleiner ist als die Zahl der Unbekannten, stellt sich schnell heraus, daß die acht Linearkombinationen der linken Seite abhängig sind. Da die rechte Seite aus beliebigen Zahlen besteht, enthält das Gleichungssystem einen Widerspruch und ist somit unlösbar. □

Hidden-Markow-Modelle (6.4.6) Hidden-Markow-Modelle sind durch eine Menge von n verdeckten Symbolen und einer Menge von m Ausgabesymbolen, einem Vektor Π von anfänglichen Wahrscheinlichkeiten und einer $n \times n$-Übergangswahrscheinlichkeitsmatrix P für die verdeckten Symbole und einer $n \times m$-Ausgabewahrscheinlichkeitsmatrix Σ definiert.

Ziel ist es, eine Zeitreihe der Länge L von Symbolen $\gamma_1, \gamma_2, \ldots, \gamma_L$ zu erklären. Die Wahrscheinlichkeit, daß die Folge $\beta_1, \ldots \beta_L$ von verdeckten Symbolen und die Folge $\gamma_1, \ldots, \gamma_L$ von Ausgabesymbolen auftaucht, ist bei Hidden-Markow-Modellen einfach gegeben durch

$$p(\beta, \gamma) = \Pi_{\beta_1} \Sigma_{\beta_1 \gamma_1} \cdot \mathrm{P}_{\beta_1 \beta_2} \Sigma_{\beta_2 \gamma_2} \ldots \mathrm{P}_{\beta_{L-1} \beta_L} \Sigma_{\beta_L \gamma_L}.$$

Die Parameter Π, P und Σ sind dann so anzupassen, daß die beobachtete Folgen γ sich gut erklären lassen. Es soll also die Wahrscheinlichkeit $\sum_\beta p(\beta, \gamma)$ in bezug auf die Parameter maximiert werden. □

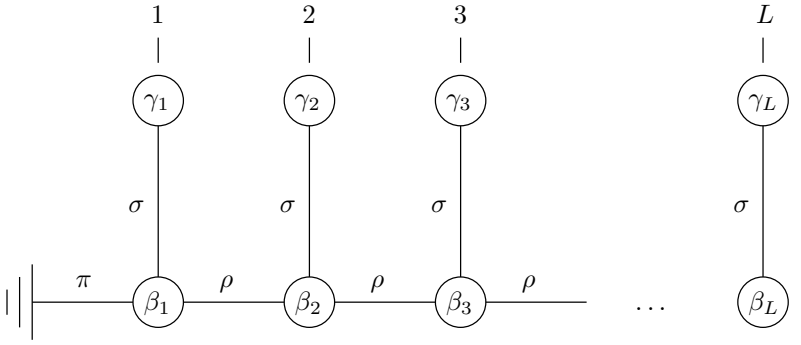


Abb. 82 Eine mehrwertige Boltzmann-Maschine, die ein in der Zeit aufgefaltetes Hidden-Markow-Modell darstellt

Zusammenhang zu Boltzmann-Maschinen (6.4.7) Abb. 82 zeigt eine gemäß (6.4.3) dezimierbare mehrwertige Boltzmann-Maschine, deren Gewichtsmatrizen miteinander gekoppelt sind. Die Energie eines Zustands $\beta\gamma$ berechnet sich zu

$$H(\pi\rho\sigma, \beta\gamma) = -\pi(1, \beta_1) - \sum_{l=1}^{L-1} \rho(\beta_l, \beta_{l+1}) - \sum_{l=1}^{L} \sigma(\beta_l, \gamma_l).$$

Die Wahrscheinlichkeit, die Boltzmann-Maschine in einem bestimmten Zustand zu treffen, ist durch

$$P_{\pi\rho\sigma}(\beta\gamma) = \frac{1}{Z} \exp(-H(\pi\rho\sigma, \beta\gamma)/T)$$

gegeben. Durch Vergleich mit (6.4.6) ist klar, daß es zu jedem Hidden-Markow-Modell mit den Parametern Π, P und Σ eine mehrwertige Boltzmann-Maschine gibt, die L n-wertige Binnenknoten, L m-wertige Ausgabeknoten und wie in Abb. 82 gekoppelte Gewichtsmatrizen π, ρ und σ, für die gilt

$$\pi(1, i) = T \log(\Pi_i),\ \rho(i, j) = T \log(\mathrm{P}_{ij}) \text{ und } \sigma(i, j) = T \log(\Sigma_{ij}).$$

Die Zustandssumme der Boltzmann-Maschine ist dann 1. Umgekehrt entspricht aber nicht jeder Gewichtsmatrixbelegung für π, ρ und σ einem Hidden-Markow-Modell, weil das dazu korrespondierende Π vielleicht kein normierter Wahrscheinlichkeitsvektor ist oder etwa P bzw. Σ keine stochastische Matrix.

Ein Hidden-Markow-Modell mit endlicher Länge L entspricht also einer mehrwertigen dezimierbaren Boltzmann-Maschine mit der speziellen, in Abb. 82 gezeichneten Struktur, gekoppelten Gewichtsmatrizen, deren Elemente selbst noch Normierungsbedingungen unterliegen. Vor allem sind die möglichen Strukturen (Leitern, Ketten, Bäume) von mehrwertigen dezimierbaren Boltzmann-Maschinen wesentlich reichhaltiger als in Abb. 82. Saul und Jordan ⟦Saul und Jordan 1995⟧ sprechen von der Möglichkeit eines „Designer-Netzes“, das z. B. langreichweitige Verbindungen einbaut. Die in Abb. 83 gegebene Interpretation zeigt also, daß dezimierbare Boltzmann-Maschinen von deutlichem Interesse sind. □

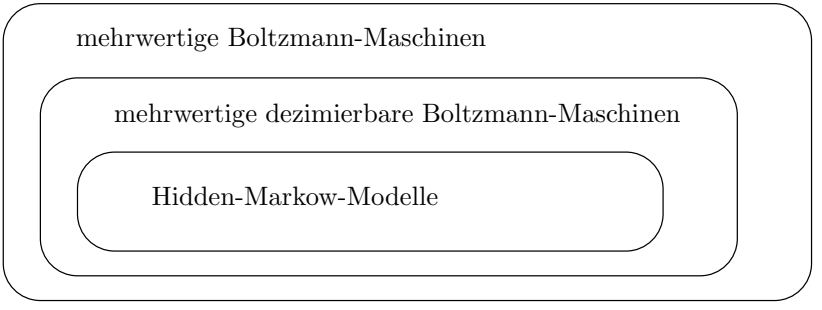


Abb. 83 Menge der mehrwertigen dezimierbaren Boltzmann-Maschinen

6.5 Effiziente Lernalgorithmen

Die Boltzmann-Lernregel basiert auf der Methode des Gradientenabstiegs. In Kapitel 2 wurde dargelegt, warum diese Methode nicht sehr effizient ist. Alle dort vorgestellten Modifikationen machen nicht nur vom Gradienten der Kostenfunktion, sondern auch von der Kostenfunktion selbst expliziten Gebrauch. Nun ist die Kostenfunktion der Eingabe-Ausgabe-Maschine der Information gain; er reduziert sich zur relativen Entropie, wenn die Menge der Eingabeknoten leer ist. In diesem Sinn ist die Eingabe-Ausgabe-Maschine die allgemeine Version der Boltzmann-Maschine. Daher ist es von Interesse, den Information gain algorithmisch effizient berechnen zu können.

Berechnung des Information gain (6.5.1) Diese Kostenfunktion ist in (5.3.4) definiert als

$$I_w = \sum_{\alpha} q(\alpha) \sum_{\gamma} r(\gamma|\alpha) \log \frac{r(\gamma|\alpha)}{p_w(\gamma|\alpha)}.$$

Dabei ist r die gewünschte gemeinsame Wahrscheinlichkeitsverteilung für Eingabe- und Ausgabeknoten, q die daraus resultierende Randverteilung für Eingabemuster und $r(\gamma|\alpha) = r(\alpha\gamma)/q(\alpha)$. $p_w(\bullet|\alpha)$ notiert die im Gleichgewicht an der Boltzmann-Maschine eingestellte bedingte Wahrscheinlichkeitsverteilung, gegeben die Eingabe α.

In der Regel ist die gewünschte Wahrscheinlichkeitsverteilung empirisch – d. h. durch m Muster (α^1, γ^1), …, (α^m, γ^m) – gegeben. Werden diese lexikographisch sortiert, so kann $q(\alpha)$ als Anzahl der Muster, die in der ersten Komponente mit α übereinstimmen, geteilt durch m festgelegt werden. In gleicher Weise bestimmt sich $r(\gamma|\alpha)$ durch die Häufigkeit des Muster (α, γ) geteilt durch $q(\alpha)$. Wegen $\lim_{x\to 0} x \log(x) = 0$ reduziert sich die Berechnung von I_w auf Summation über *in der Datenmenge vorkommende* Muster (α, γ). Es bleibt $\log(p_w(\gamma|\alpha))$ zu berechnen, was aber mit den Techniken des Abschnitts 6.3 nicht schwer fällt.

Manchmal, z. B. bei der Dezimierung der Binnenknoten (6.3.1), der direkten Berechnung (6.3.2) und der zweifachen Dezimierung (6.3.4) spaltet sich $\log(p_w(\gamma|\alpha))$ in einen nur von α abhängigen Teil und einen von γ und α abhängigen Teil auf. Dann kann der nur von α abhängige Teil natürlich aus der inneren Summe zur Berechnung des Information gain herausgezogen werden. Hier ergibt sich dann für binnendezimierbare Boltzmann-Maschine eine besonders schnelle Berechnung des Information gain. □

Partielle Ableitungen (6.2.14) der Zustandssumme ergeben die wesentlichen Bausteine $\mathrm{E}(s_a s_b)_\delta$ für den Gradient der Kostenfunktion, der durch die Lernregel (5.3.3) gegeben ist. Obiger Abschnitt zeigt, wie die Kostenfunktion exakt und effizient zu berechnen ist. Dies kann in Kombination mit Versionen der stabilen Parameteradaption aus 2.3 für konkrete Lernalgorithmen benutzt werden. Der Vorteil der stabilen Parameteradaptionen ist, daß sie mit sehr wenigen Auswertungen der Kostenfunktion auskommen.

6.6 Eine Anwendung

In diesem Abschnitt sollen die vorangegangenen Techniken benutzt werden, um ein kleines medizinisches Diagnoseproblem zu lösen, das in ⟦Lauritzen und Spiegelhalter 1988⟧ als didaktisches Beispiel verwendet wird. Dazu wird zunächst das wahrscheinlichkeitstheoretische Modell der Belief-Netze ⟦z. B. Neapolitan 1990⟧ eingeführt. Mit diesen wird ein primitives Modell des medizinischen Problems der Dyspnœ (Atemnot) aufgestellt. Daten, die von diesem Modell erzeugt wurden, werden danach zum Trainieren von einigen Boltzmann-Maschinen verwendet. Mit Hilfe einer Verallgemeinerung der im vierten Kapitel entwickelten Technik der robusten Modellauswahl wird eine sinnvolle dezimierbare Boltzmann-Maschine eingesetzt, um anhand von Beispielen das statistische Expertenwissen über das Dyspnœ-Beispiel zu speichern.

In ⟦Rüger et al. 1996⟧ wurden bereits unbeschränkte Boltzmann-Maschine untersucht und ihre Ergebnisse mit denen von dezimierbaren Boltzmann-Maschine verglichen. Dabei zeigte sich, daß letztere in der Tat nicht nur völlig geeignet sind, das Diagnoseproblem zu lösen, sondern dies auch in erheblich kürzerer Zeit tun. Das trifft nicht nur auf die verbrauchte Rechenzeit zum Lernen und Schließen zu, sondern in besonderem Maß auf die immense Experimentatorzeit zum Einstellen der optimalen Parameter bei herkömmlichem Lernen (Annealing schedule und Parameter zum Ziehen der Stichprobe beim Ausnutzen der Ergodentheorie für das herkömmliche Lernen (5.2.5)). Daher beschränkt sich die folgende Darstellung auf die Untersuchung dezimierbarer Boltzmann-Maschinen.

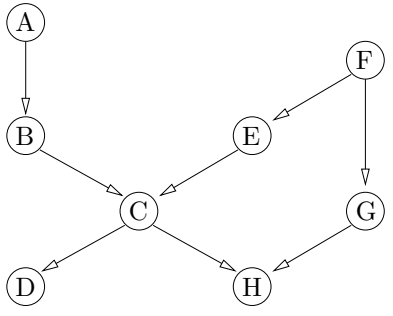
Belief-Netze (6.6.1) Ein *Belief-Netz* (N, E, X, P_X) ist ein Feedforward-Netz (N, E) zusammen mit einer Zufallsvariable X_a pro Knoten a, wenn für die gemeinsame Wahrscheinlichkeitsverteilung P_X von X gilt

$$P_X(x) = \prod_{a \in N} P_X(X_a = x_a | X_{L_a} = x_{L_a}).$$

L_a ist dabei die in (1.1.5) definierte Menge der Vorgängerknoten eines Knotens a. □

Bemerkung (6.6.2) Die gemeinsame Wahrscheinlichkeitsverteilung ist in Belief-Netzen also schon durch die bedingten Wahrscheinlichkeiten eines jeden Knotens bei Realisierungen der Zufallsvariablen der Vorgängerknoten festgelegt. Diese Struktur vereinfacht die Angabe der gemeinsamen Wahrscheinlichkeitsverteilung erheblich. □

Ein didaktisches Beispiel (6.6.3) Abb. 84 aus [[Lauritzen und Spiegelhalter 1988]] stellt ein Beispiel für die Anwendung von Belief-Netzen dar. Dabei werden acht zweiwertige Zufallsvariablen in einen kausalen Zusammenhang miteinander gebracht: Tuberkulose und Lungenkrebs können einen positiven Röntgenbefund nach sich ziehen; beides kann aber auch Dyspnœ verursachen. Die wiederum kann auch von Bronchitis hervorgerufen werden. Ein kürzlicher Besuch in Asien erhöht die Wahrscheinlichkeit für Tuberkulose, während Rauchen eine mögliche Ursache für Lungenkrebs oder Bronchitis ist. Durch die Angabe von bedingten Wahrscheinlichkeiten kann nun ein plausibles, wenn auch primitives, wahrscheinlichkeitstheoretisches Modell für Dyspnœ aufgestellt werden.



A: Asienbesuch
B: Tuberkulose
C: E oder B
D: pos. Röntgenbefund
E: Lungenkrebs
F: Raucher
G: Bronchitis
H: Dyspnœ

Abb. 84 Belief-Netz für Dyspnœ

Um ein konkretes Beispiel zu haben, wurden folgende Zuordnungen gewählt (dabei bedeute y, daß die Zufallsvariable Y in der Bedeutung von „Y ist vorhanden“ realisiert wurde): $P(a) = 0{,}01$, $P(b|a) = 0{,}05$, $P(b|\neg a) = 0{,}01$, $P(c|b,e) = 1$, $P(c|b,\neg e) = 1$, $P(c|\neg b,e) = 1$, $P(c|\neg b,\neg e) = 0$, $P(d|c) = 0{,}98$, $P(d|\neg c) = 0{,}05$, $P(e|f) = 0{,}1$, $P(e|\neg f) = 0{,}01$, $P(f) = 0{,}5$, $P(g|f) = 0{,}6$, $P(g|\neg f) = 0{,}3$, $P(h|c,g) = 0{,}9$, $P(h|c,\neg g) = 0{,}7$, $P(h|\neg c,g) = 0{,}8$, $P(h|\neg c,\neg g) = 0{,}1$

Diese achtzehn Zahlen legen gemäß (6.6.1) bereits die gemeinsame Wahrscheinlichkeitsverteilung fest, wozu sonst $255 = 2^8 - 1$ Zahlen nötig wären. □

Die Boltzmann-Maschine als statistisches Modell (6.6.4) Belief-Netze werden konstruiert; sie drücken die Vorstellung dessen aus, der sie erzeugt hat. In vielen Fällen ist eine innere Einsicht in die stochastische Struktur der betrachteten Zufallsvariablen nicht a priori gegeben. Gemäß dem Programmierparadigma künstlicher neuronaler Netze sollte ein System anhand von Beispielen den Zusammenhang lernen. Zu diesem Zweck wurden mit dem Belief-Netz (6.6.3) zunächst Realisierungen der Zufallsvariable $X = (A, B, \ldots, H)$ synthetisch erzeugt. Diese dienen als „diagnostizierte Fälle".

Eine Boltzmann-Maschine soll dann nur anhand dieser Beispiele die gemeinsame Wahrscheinlichkeitsverteilung P_X von X approximieren. In einer realistischen Anwendung würde man die Zufallsvariablen A, D, F und H zu Eingabeknoten assoziieren, weil sie leicht festzustellen sind. Die möglichen Diagnosen B, E und G hingegen wären als Ausgabeknoten zu modellieren. C ist weder ein Symptom noch eine eigenständige interessante Zufallsvariable; sie wird nicht explizit modelliert (d. h. die Boltzmann-Maschine soll auch nur die interessierende Randverteilung von P_X bezüglich C approximieren).

Binnenknoten vermitteln die Abhängigkeiten der interessierenden Zufallsvariablen untereinander. Die Anzahl der Binnenknoten und ihre Verbindungsstruktur ist nicht von vornherein klar. Dies festzulegen ist Gegenstand einer Modellauswahl, wie sie im folgenden beschrieben wird.

Mit einem Lernalgorithmus aus 6.5 können dann die Gewichte der Boltzmann-Maschine angepaßt werden, um an den sichtbaren Knoten möglichst gut die relevante Wahrscheinlichkeitsverteilung wiedergeben zu können. Der fertig trainierten Boltzmann-Maschine kann im Anwendungsfall alles über die Situation Bekannte, sagen wir α, aufgeprägt werden. Eine beliebige Kombination von anderen interessierenden Knoten – z. B. die zu E und G assoziierten – kann als Ausgabeschicht O gewählt werden. Dann kann für jede Belegung γ dieser Knoten, wie etwa „G, und nicht E, liegt

vor“, die Wahrscheinlichkeit $p_w(\gamma|\alpha)$ gemäß den Techniken des effizienten Schließens aus Abschnitt 6.3 berechnet werden. □

Modellauswahl für Boltzmann-Maschinen (6.6.5) In Abschnitt 4.6 wurde gezeigt, wie Clustern im effektiven Gewichtungsraum von Feedforward-Netzen möglich ist. Nun haben vollverknüpfte Boltzmann-Maschinen dieselben Symmetrien wie die in 4.6 behandelten Netze: Zwei Binnenknoten können vertauscht werden, ohne daß dies etwas an der Wahrscheinlichkeitsverteilung an den sichtbaren Knoten ändert. Genauso können alle Gewichtungskomponenten eines Binnenknoten ihr Vorzeichen ändern ohne Wirkung auf die sichtbaren Knoten: dies liegt an der Symmetrie der Aktivationen $\{-1, 1\}$ und der Punktsymmetrie der Fermifunktion zur Bestimmung der Wahrscheinlichkeit der Aktivationen (5.1.1).

Bei nicht vollverknüpften Modellen E bietet sich dann an, das umfassendere vollverknüpfte Modell E^* zu betrachten und die nicht benötigten Komponenten des Gewichtungsvektors auf Null zu setzen. Sei $^*\colon \mathbb{R}^E \to \mathbb{R}^{E^*}$ diese Einbettung $w \mapsto w^*$ eines Gewichtungsvektors in den höherdimensionalen Gewichtungsraum. * ist eine Isometrie und über

$$d(v, w) := \min_{s \in S^*} d_E(s(v^*), w^*)$$

wird wie in (4.4.4) ein kanonischer Abstand für $v, w \in \mathbb{R}^E$ festgelegt; dabei ist S^* die natürliche Symmetriegruppe in $\mathbb{R}^{E^*}$. Mit dem so festgelegten Abstand kann also auch bei Boltzmann-Maschinen die geometrische Lageinformation der resultierenden Gewichtungsvektoren genutzt werden. □

Bemerkung (6.6.6) Obige Definition eines kanonischen Abstands kann genauso in mehrschichtigen Netzen mit gekoppelten Gewichten, Shortcuts u. ä. gegeben werden. Die Behandlung der Symmetrien in Kapitel 4 ist also wesentlich allgemeiner als es zunächst dort den Anschein haben mochte. □

Anwendung der robusten Modellauswahl (6.6.7) Gemäß (6.6.4) wurden synthetische Trainingsbeispiele für das Dyspnœ-Problem erzeugt. Viele Boltzmann-Maschinen mit vier Eingabe- und drei Ausgabeknoten

wurden damit jeweils mehrfach trainiert, wobei die anfänglichen Gewichtungsvektoren zufällig waren. Pro Modell wurden die resultierenden Gewichtungsvektoren mit der Metrik aus (6.6.5) geclustert und pro Cluster eine robuste Statistik der Kostenfunktion erstellt. Dieser Wert spiegelt die Güte des Modells für die Fähigkeit, die Beispielsdaten zu verstehen, wider. Tabelle 6 zeigt das Ergebnis für zwölf Modelle: Die Modelle D_i sind in Abb. 78 gezeichnet und D_i-v entspricht dem Modell D_i, nur daß bei D_i-v die Eingabeknoten mit allen anderen Knoten verknüpft sind. Es haben sich bei den zwölf Versuchen siebzehn Cluster gebildet, deren typischer Wert der Kostenfunktion nach dem Lernen in Tabelle 6 gelistet ist. Offensichtlich ist das Modell D_4-v besonders gut geeignet. □

Tabelle 6 Pro-Cluster-Statistik der Kostenfunktion

Netz	N/m	Median	Netz	N/m	Median
D_4-v	97%	0,0086	D_2	72%	0,0187
D_2-v	86%	0,0096	D_3	81%	0,0191
D_4	82%	0,0098	D_5	14%	0,0230
D_5-v	86%	0,0099	D_3	15%	0,0242
D_6	69%	0,0103	D_5	58%	0,0261
D_3-v	86%	0,0107	D_5	24%	0,1644
D_6-v	74%	0,0111	D_2	14%	0,2864
D_1-v	84%	0,0120	D_1	73%	0,2941
D_6-v	12%	0,0172			

Anwendung des Schließens (6.6.8) Nachdem das Modell D_4-v mit den hypothetischen Beispielsdaten aus (6.6.3) trainiert wurde, stellt sich die Frage, wie gut diese Boltzmann-Maschine nun konkret Schlüsse aus vorgegebenen unvollständigen Daten ziehen kann. Für die gemäß der Verteilung von $Y = (A, B, D, \ldots, H)$ erzeugte Trainingsmultimenge (die redundante Variable C wurde ja nicht modelliert) wurde jeweils die Eingabevariable $\alpha = (a, d, f, h)$ aufgeprägt und die Wahrscheinlichkeit $p_w(\gamma|\alpha)$ mit $\gamma = (b, e, g)$ berechnet. Das Histogramm in Abb. 85 zeigt für dieses Beispiel die Abweichungen von der durch die Beispiele vorgegebenen bedingten

Wahrscheinlichkeit $r(\gamma|\alpha)$. Jedes Trainingsmuster taucht im Histogramm mit seiner Multiplizität gewichtet auf.

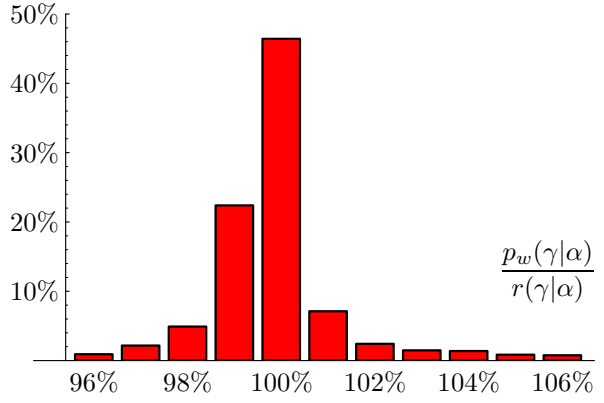


Abb. 85 Histogramm des relativen Fehlers beim Schließen

Nun ist beim Schließen die Ausgabemenge ja frei und statt der bedingten gemeinsamen Wahrscheinlichkeit von (b, e, g) kann auch jeweils die bedingte Randwahrscheinlichkeit von z. B. nur b, e oder g berechnet werden. In Tabelle 7 sind einige konkrete Daten aufgelistet.

Tabelle 7 Einige bedingte Wahrscheinlichkeiten im Vergleich

α	γ	$r(\gamma\|\alpha)$	$p_w(\gamma\|\alpha)$
$(\neg a, \neg d, \neg f, h)$	b	$< 0{,}001$	$< 0{,}001$
	e	$< 0{,}001$	$< 0{,}001$
	g	$0{,}774$	$0{,}775$
$(\neg a, \neg d, f, h)$	b	$< 0{,}001$	$< 0{,}001$
	e	$0{,}004$	$0{,}003$
	g	$0{,}922$	$0{,}922$
$(\neg a, d, f, h)$	b	$0{,}073$	$0{,}088$
	e	$0{,}726$	$0{,}731$
	g	$0{,}714$	$0{,}715$

Jeweils drei Zeilen gehören zur gleichen Eingabe. Die letzten beiden Zeilen der Tabelle deuten eine vielleicht notwendige Differentialdiagnose zwischen e und g an. Die vom Netz gegebenen Antworten sind ganz ordentlich im Vergleich zu den erwarteten. Die Boltzmann-Maschine wurde nur 40 Epochen mit der stabilen Parameteradaption (2.4.10) trainiert, und eine bessere Approximation wäre leicht möglich gewesen. Das gilt auch für das Histogramm in Abb. 85. □

6.7 Wertung

In diesem Kapitel wurde die Technik der Dezimierung erschöpfend angewendet: Für sowohl traditionelle als auch mehrwertige Boltzmann-Maschinen konnte die Menge der dezimierbaren Boltzmann-Maschinen genau charakterisiert werden. Die mehrwertigen dezimierbaren Boltzmann-Maschinen erweisen sich als eine reichhaltige Verallgemeinerung von Hidden-Markow-Modellen. Letztere werden erfolgreich bei Problemen neuronalen Lernens – wie der Sprachverarbeitung – eingesetzt; schon alleine deswegen lohnt sich das Studium der Boltzmann-Maschine.

Ich habe durch Anwendung der Fourier-Stieltjes-Transformation gezeigt, daß die Gewichtsabhängigkeit der Zustandssumme einer allgemeinen Boltzmann-Maschine eine Schlüsselrolle spielt. Damit können in dezimierbaren Boltzmann-Maschinen nicht nur Lernregeln berechnet, sondern auch das gespeicherte statistische Wissen abgerufen werden. Für beide Fälle habe ich die relevanten Algorithmen beschrieben: Eine außerordentlich effiziente Methode zum Berechnen der Wechselwirkungsterme $\mathrm{E}(s_a s_b) = 2T\partial \log(Z_w)/\partial w_{ab}$ ist es, das Abbildungsnetz mit der Netzabbildung $w \mapsto \log(Z_w)$ zu konstruieren, das durch Dezimierung entsteht. Die Kettenregel in Abbildungsnetzen erlaubt dann, durch einmaliges Vorwärts- und Rückwärtspropagieren gleichzeitig alle Wechselwirkungsterme zu berechnen. Für das Schließen in dezimierbaren Boltzmann-Maschinen habe ich mehrere Varianten angegeben, die jeweils die Zustandssumme benutzen.

In diesem Sinn kann das Problem des Lernens und Schließens in einer dezimierbaren Boltzmann-Maschine transformiert werden auf das Feedforward-Netz zur Berechnung der Zustandssumme. Insbesondere sind dabei

alle Berechnungen exakt – im Gegensatz zum herkömmlichen Schätzen der Wechselwirkung $\mathrm{E}(s_a s_b)$ mit Hilfe der Ergodentheorie. Die Lernregel für Boltzmann-Maschinen ist durch Gradientenabstieg in einer Kostenfunktion gegeben. Mit den erwähnten Techniken ist es nicht nur möglich, den Gradienten der Kostenfunktion für dezimierbare Boltzmann-Maschinen exakt zu berechnen, sondern auch die Kostenfunktion selbst; daher lassen sich auch sämtliche Beschleunigungsmethoden des zweiten Kapitels anwenden.

Des weiteren ist es mir gelungen, die im zweiten Kapitel entwickelte Modellbewertung durch Clustern im Gewichtungsraum auf Boltzmann-Maschinen anzuwenden. Es zeigt sich, daß die ursprünglich zum Clustern im Gewichtungsraum von Feedforward-Netzen gemachte kanonische Metrik auf allgemeine Netze – also auch auf Boltzmann-Maschinen – übertragbar ist.

Kurz: *Dezimierbare Boltzmann-Maschinen haben alle Vorzüge von rekurrenten und stochastischen Netzen – und das zu den Kosten eines deterministischen Feedforward-Netzes!* Eine kleine Anwendung aller Techniken an einem konkreten Beispiel hat gezeigt, daß sich die theoretischen Zusammenhänge problemlos in die Praxis umsetzen lassen.

Ausblick

... to boldly go where no one has gone before®

— Paramount Pictures

Hier werden die Hauptergebnisse der einzelnen Kapitel dargestellt, und es wird ein Ausblick auf die Forschungsarbeiten gegeben, von denen angenommen werden kann, daß sie in naher Zukunft Früchte tragen werden.

Die ersten vier Kapitel dieser Arbeit behandeln wesentliche theoretische Grundfragen des überwachten Lernens in künstlichen neuronalen Netzen. Sie beginnt mit dem Studium dessen, was allen neuronalen Berechnungsmodellen gemein ist: gerichtete Graphen von einfachen Berechnungselementen; diese werden für Feedforward- und rekurrente Netze diskutiert. Die Frage, *wie neuronale Netze überhaupt trainiert werden können*, wird im Einklang mit der Lehrbuchliteratur auf das Bilden des Credit assignments für die beteiligten Gewichte zurückgeführt. Im Gegensatz zur Lehrbuchliteratur wird die Berechnung des Credit assignments nicht für jedes Lernverfahren einzeln vorgenommen, sondern einmal systematisch mit Hilfe der hier formulierten Kettenregel in allgemeinen Abbildungsnetzen, die zwei natürliche algorithmische Varianten besitzt.

Zur Frage, *wie neuronale Netze effizient trainiert werden können*, habe ich eine eigene Klasse von Lernverfahren beigetragen, die (asymptotisch) stabilen Parameteradaptionen. Dies wurde motiviert durch die im Tradeoff-Theorem formulierten Unzulänglichkeiten des wohl populärsten Lernverfahrens Backpropagation. Durch Adaption des Lernparameters konnten diese ausgeräumt und die Konvergenz der Lernverfahren bewiesen werden. Nun haben andere Lernverfahren auch Parameter: Standard-Kohonen-Lernen hat neben der Lernrate einen Zerfallsparameter, genetische Algorithmen haben Wahrscheinlichkeiten der Selektions- und Mutationsoperatoren usw. Hier sollte in Zukunft untersucht werden, wie sich die stabile Parameteradaption übertragen läßt. Die Wahl der Suchrichtung ist bei der stabilen Parameteradaption weitgehend frei. Ich habe nachgewiesen, daß deutliche Geschwindigkeitssteigerungen durch den Einsatz von Polak-Ribière-Richtungen zustande kommen. Dies ist aber nicht die einzige Möglichkeit,

Zweite-Ordnung-Information einfließen zu lassen. Es bietet sich etwa an, in Zukunft auch andere Suchrichtungen wie z. B. die des Broyden-Fletcher-Goldfarb-Shanno-Algorithmus ⟦Press et al. 1988⟧ in Betracht zu ziehen und zu studieren.

Eine weitere wichtige Frage, *wie die Qualität der Netzfunktion eines trainierten Netzes bewertet werden kann*, wurde gestellt und einige bekannte Antworten der Statistik vorgestellt. Dazu ist es zunächst nötig, das neuronale Netz als Regressionsmodell zu betrachten. Es zeigt sich, daß Unzulänglichkeiten neuronalen Lernens, wie das Steckenbleiben in lokalen Minima, herausgefiltert werden müssen.

Dies gelingt durch eine zusammen mit Dr. Ossen neu entwickelte Methode der Analyse des Gewichtungsraums. Von unserem daraus entstandenen neuen Verfahren Clustered bootstrap haben wir bei Regressionsmodellen und sogar bei der Zeitreihenanalyse demonstriert, daß es erfolgreich in der Praxis eingesetzt werden kann. Mit der Analyse des Gewichtungsraums kann auch die Frage beantwortet werden, *wie ein Netzmodell bewertet bzw. ausgewählt werden kann*, das die vorgegeben Daten erklärt. Unsere – auch ohne Annahme von zufälligen Einflüssen in den Daten anwendbare – Methode der robusten Modellauswahl wurde anhand von synthetischen Beispielsdaten demonstriert. Der nächste Schritt sollte nun sein, diese Methode an wirklichen technischen Problemen zu testen und zu beurteilen. Dies wird im Augenblick leider dadurch erschwert, daß der objektiven Beurteilung von Verfahren zur Modellauswahl bisher wenig Aufmerksamkeit geschenkt wurde; es scheint noch keine etablierte Testumgebung für solche Probleme zu geben.

Die letzten beiden Kapitel dieser Arbeit behandeln ein interessantes Netzmodell, das vielleicht deswegen bisher sowenig Beachtung fand, weil es als langsam und ineffizient gilt: die Boltzmann-Maschine. Zu den Knoten dieses Netzes sind Zufallsvariablen assoziiert, deren gemeinsame Verteilung bereits in der Thermodynamik eine wichtige Rolle spielt. Durch den Einsatz von Binnenknoten kann die Boltzmann-Maschine zur Approximation von Wahrscheinlichkeitsverteilungen benutzt werden. Im allgemeinen erweist sich dieses Vorgehen als intraktabel.

In speziellen, dünn vernetzten Boltzmann-Maschine ist aber dieses allgemeine Problem nicht gegeben. Mit Hilfe der aus der statistischen Mechanik bekannten Technik der Dezimierung konnte ich zu jeder dezimierbaren Boltzmann-Maschine ein zugehöriges deterministisches Feedforward-Netz konstruieren, von dem ich gezeigt habe, wie damit alle für das Lernen und Schließen relevanten Größen exakt berechnet werden können. Eine auf der Hand liegende zukünftige Erweiterung ist, mehrwertige dezimierbare Boltzmann-Maschine auf die gleiche Weise zu behandeln. Die so entstehenden Netzmodelle und Algorithmen verallgemeinern wesentlich die Hidden-Markow-Modelle. Deswegen sind interessante Anwendungen der mehrwertigen dezimierbaren Boltzmann-Maschinen z. B. in der Sprachverarbeitung zu erwarten.

Dezimierbare Boltzmann-Maschinen können eingesetzt werden zur Approximation von Wahrscheinlichkeitsverteilungen, die alleine durch Beispiele gegeben sind. Fehlende Kanten in dem Netz bedeuten eine bestimmte bedingte Unabhängigkeit in der resultierenden Verteilung. Eine konkrete dezimierbare Boltzmann-Maschine kann also offensichtlich nicht alle Verteilungen annähern. Eine offene Frage hierbei ist aber, ob immer zusätzliche dünn verknüpfte Binnenknoten eingesetzt werden können, so daß die Boltzmann-Maschine dezimierbar bleibt und so daß eine durch Beispiele gegebene Zielverteilung besser approximiert werden kann. Eine Bejahung dieser Frage wäre eine analoge Version des Theorems von Cybenko für Boltzmann-Maschinen. Dies würde helfen, die Bewertung von Modellen mit Hilfe der Clusteranalyse zu einer erfolgreichen Modellauswahl zu machen.

Diese und andere Fragen bleiben für weitere Forschungsarbeiten.

Literaturverzeichnis

A couple of months in the laboratory can frequently save a couple of hours in the library.

— unbekannte Quelle

[[Ackley et al. 1985]] D. H. Ackley, G. E. Hinton und T. J. Sejnowski: *A Learning Algorithm for Boltzmann Machines*. Cognitive Science 9, 147–169

[[Alspector et al. 1992]] J. Alspector, A. Jayakumar und S. Luna: *Experimental Evaluation of Learning in a Neural Microsystem*. In [[NIPS 4]], 871–878

[[Bauer 1990]] H. Bauer: *Maß- und Integrationstheorie*. Berlin: de Gruyter

[[Bauer 1991]] H. Bauer: *Wahrscheinlichkeitstheorie*. Berlin: de Gruyter

[[Baum 1972]] L. Baum: *An Inequality and Associated Maximization Technique in Statistical Estimation of Probabilistic Functions of Markov Processes*. Inequalities 3, 1–8

[[Bishop 1995]] C. M. Bishop: *Neural Networks for Pattern Recognition*. Oxford University Press

[[Bryson und Ho 1969]] A. E. Bryson und Y.-C. Ho: *Applied Optimal Control*. New York: Blaisdell Publishing Co.

[[Buntine und Weigend 1994]] W. L. Buntine und A. S. Weigend: *Computing Second Derivatives on Feed-Forward Networks: A Review*. IEEE Transactions on Neural Networks 5 (3), 480–488

[[Callen 1985]] H. B. Callen: *Thermodynamics and an Introduction to Thermostatistics*. New York: John Wiley & Sons

[[Chen et al. 1993]] A. M. Chen, H. Lu und R. Hecht-Nielsen: *On the Geometry of Feedforward Neural Network Error Surfaces*. Neural Computation 5 (6), 910–927

[[CMSS-88]] D. S. Touretzky, G. E. Hinton und T. J. Sejnowski (Hrsg.): *Proceedings of the 1988 Connectionist Models Summer School*. San Mateo: Morgan Kaufmann Publishers 1989

[[Cohen 1970]] A. Cohen: *Rate of Convergence for Root Finding and Optimization Algorithms*. University of California Berkeley: Dissertation

[[COLT-89]] R. Rivest, D. Haussler und M. Warmuth (Hrsg.): *COLT 89*. San Mateo: Morgan Kaufmann Publishers 1989

[[Cooper 1990]] G. F. Cooper: *The Computational Complexity of Probabilistic Inference Using Bayesian Belief Networks*. Artificial Intelligence 42, 393–405

[[CoWAN-96]] Fachbereich Mathematik (Hrsg.): *CoWAN'96 Tagungsband*. Brandenburgische Technische Universität Cottbus 1996

[[Cybenko 1989]] G. Cybenko: *Approximation by Superpositions of a Sigmoidal Function*. Mathematics of Control, Signals, and Systems 2, 303–314

[[Dagnum und Luby 1993]] P. Dagnum und M. Luby: *Approximating probabilistic inference in Bayesian belief networks is NP-hard*. Artificial Intelligence 60, 141–153

[[Deco und Obradovic 1996]] G. Deco und D. Obradovic: *An Information-Theoretic Approach to Neural Computing*. New York: Springer-Verlag

[[Domb und Green 1972]] C. Domb und M. S. Green (Hrsg.): *Phase Transitions and Critical Phenomena. Volume 1. Exact results*. London: Academic Press

[[Duda und Hart 1973]] R. O. Duda und P. E. Hart: *Pattern classification and scene analysis*. New York: John Wiley & Sons

[[Efron und Tibshirani 1993]] B. Efron und R. J. Tibshirani: *An Introduction to the Bootstrap*. New York: Chapman & Hall

[[Eggarter 1974]] T. P. Eggarter: *Cayley trees, the Ising problem, and the thermodynamic limit*. Physical Review B 9 (7), 2989–2992

[[ESANN-95]] M. Verleysen (Hrsg.): *European Symposium on Artificial Neural Networks*. Brüssel: D facto publications 1996

[[Fahlman 1989]] S. E. Fahlman: *Fast-Learning Variations on Back-Propagation: An Empirical Study.* In [[CMSS-88]], 38–51

[[Falk und Ruppel 1976]] G. Falk und W. Ruppel: *Energie und Entropie. Eine Einführung in die Thermodynamik.* Berlin: Springer-Verlag

[[Feller 1971]] W. Feller: *An Introduction to Probability Theory and Its Applications, Vol. II.* New York: John Wiley & Sons

[[Finkelstein 1988]] D. Finkelstein: *Finite Physics.* In [[Herken 1988]]

[[Freeman und Skapura 1991]] J. A. Freeman und D. M. Skapura: *Neural Networks. Algorithms, Applications, and Programming Techniques.* Redwood City: Addison-Wesley

[[Frieden 1991]] B. R. Frieden: *Probability, Statistical Optics, and Data Testing.* Berlin: Springer-Verlag

[[Garey und Johnson 1979]] M. R. Garey und D. S. Johnson: *Computers and Intractability; a Guide to the Theory of NP-Completeness.* New York: W. H. Freeman and Co.

[[Hadamard 1945]] J. Hadamard: *The Psychology of Invention in the Mathematical Field.* Princeton University Press

[[Hanson und Pratt 1989]] S. J. Hanson und L. Y. Pratt: *Comparing Biases for Minimal Network Construction with Back-Propagation.* In [[NIPS 1]], 177–185

[[Hassibi und Stork 1993]] B. Hassibi und D. G. Stork: *Second order derivatives for network pruning: Optimal Brain Surgeon.* In [[NIPS 5]], 164–171

[[Hebb 1949]] D. O. Hebb: *The Organization of Behavior.* New York: John Wiley & Sons

[[Hecht-Nielsen 1989]] R. Hecht-Nielsen: *Neurocomputing.* Redwood City: Addison-Wesley

[[HeKoNN-94]] I. Duwe, F. Kurfeß, G. Paaß und S. Vogel (Hrsg.): *Konnektionismus und Neuronale Netze.* Sankt Augustin: GMD 1994

[[Herken 1988]] R. Herken (Hrsg.): *The Universal Turing Machine. A Half-Century Survey.* Hamburg: Kammerer & Unverzagt

[[Hertz et al. 1991]] J. Hertz, A. Krogh und R. G. Palmer: *Introduction to the Theory of Neural Computation*. Redwood City: Addison-Wesley

[[Heskes und Kappen 1993]] T. M. Heskes und H. J. Kappen: *On-line learning processes in artificial neural networks*. In [[Taylor 1993]], 199–233

[[Hinton und Sejnowski 1983]] G. E. Hinton und T. J. Sejnowski: *Optimal Perceptual Inference*. In „Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (Washington 1983)“, 448–453, New York: IEEE

[[Hinton und Sejnowski 1986]] G. E. Hinton und T. J. Sejnowski: *Learning and Relearning in Boltzmann Machines*. In [[Rumelhart et al. 1986b]], 282–317

[[Hopfield et al. 1983]] J. J. Hopfield, D. I. Feinstein und R. G. Palmer: *“Unlearning” Has a Stabilizing Effect in Collective Memories*. Nature 304, 158–159

[[Hornik et al. 1989]] K. Hornik, M. B. Stinchcombe und H. White: *Multilayer Feedforward Networks are Universal Approximators*. Neural Networks 2, 359–366

[[ICANN-96]] C. von der Malsburg, W. von Seelen, J. C. Vorbrüggen und B. Sendhoff (Hrsg.): *Artificial Neural Networks*. Berlin: Springer-Verlag 1996, Lecture Notes in Computer Science 1112

[[ICNC-90]] R. Eckmiller, G. Hartmann und G. Hauske (Hrsg.): *Parallel Processing in Neural Systems and Computers*. Amsterdam: Elsevier Science Publishers 1990

[[IJCNN-93]] *Proceedings of the IJCNN ’93, Nagoya, Japan.*

[[Itzykson und Drouffe 1991]] C. Itzykson und J. Drouffe: *Statistical Field Theory*. Cambridge University Press

[[Jacobs 1977]] D. A. H. Jacobs (Hrsg.): *The State of Art in Numerical Analysis*. London: Academic Press

[[Jordan und Rumelhart 1990]] M. I. Jordan und D. E. Rumelhart: *Forward models: Supervised learning with a distal teacher*. MIT Center for Cognitive Science: Occasional Paper no. 40

[[Juang und Rabiner 1991]] B. H. Juang und L. R. Rabiner: *Hidden Markov Models for Speech Recognition*. Technometrics 33, 251–272

[[Kockelkorn 1997]] U. Kockelkorn: *Lineare Modelle*. Oldenburg-Verlag

[[Kok 1996]] J. N. Kok: Vortrag und Diskussion zu [[Kok et al. 1996]]

[[Kok et al. 1996]] J. N. Kok, E. Marchiori, M. Marchiori und C. Rossi: *Constraining of Weights Using Regularities*. In [[ESANN-95]], 267–272

[[Krengel 1991]] U. Krengel: *Einführung in die Wahrscheinlichkeitstheorie und Statistik*. Braunschweig: Vieweg

[[Kullback 1959]] S. Kullback: *Information Theory and Statistics*. New York: John Wiley & Sons

[[Lau 1992]] C. G. Lau (Hrsg.): *Neural Networks. Theoretical Foundations and Analysis*. New York: IEEE

[[Lauritzen und Spiegelhalter 1988]] S. L. Lauritzen und D. J. Spiegelhalter: *Local Computations with Probabilities on Graphical Structures and their Application to Expert Systems* (mit Diskussion). Journal of the Royal Statistical Society B 50, 157–224

[[Lawler et al. 1985]] E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan und D. B. Shmoys (Hrsg.): *The Traveling Salesman Problem*. Chichester: John Wiley & Sons

[[le Cun 1989a]] Y. le Cun: *A Theoretical Framework for Back-Propagation*. In [[CMSS-88]], 21–28

[[le Cun 1989b]] Y. le Cun: *Generalization and Network Design Strategies*. In [[Pfeifer et al. 1989]], 143–155

[[Lin und Vitter 1989]] J.-H. Lin und J. S. Vitter: *Complexity Issues in Learning by Neural Nets*. In [[COLT-89]], 118–132

[[Macdonald 1975]] I. D. Macdonald: *The theory of groups*. Oxford University Press

[[MacKay 1996]] D. J. C. MacKay: *Equivalence of Boltzmann Chains and Hidden Markov Models*. Neural Computation 8 (1), 178–181

[[McClelland und Rumelhart 1988]] J. L. McClelland und D. E. Rumelhart: *Parallel Distributed Processing: A Handbook of Models, Programs, and Exercises*. Cambridge: MIT Press

⟦Metropolis et al. 1953⟧ N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller und E. Teller: *Equation of State Calculations for Fast Computing Machines*. Journal of Chemical Physics 21, 1087–1092

⟦Moreira und Fiesler 1995⟧ M. Moreira und E. Fiesler: *Neural Networks with Adaptive Learning Rate and Momentum Terms*. Institut Dalle Molle d'Intelligence Artificielle Perceptive, Martigny: Technical report 95-04

⟦Müller und Reinhardt 1990⟧ B. Müller und J. Reinhardt: *Neural Networks. An Introduction*. Berlin: Springer-Verlag

⟦Murray et al. 1994⟧ M. Murray, M.-T. Leung, K. Boonyanit, K. Kritayakirana, J. B. Burr, G. J. Wolff, T. Watanabe, E. Schwartz, D. G. Stork und A. M. Peterson: *Digital Boltzmann VLSI for constraint satisfaction and learning*. In ⟦NIPS 6⟧, 896–903

⟦Neapolitan 1990⟧ R. E. Neapolitan: *Probabilistic Reasoning in Expert Systems. Theory and Algorithms*. New York: John Wiley & Sons

⟦NEURAP-95⟧ N. Giambiasi, J. C. Bertrand, C. Frydman und D. Bertrand (Hrsg.): *Neural Networks and Their Applications*. Marseille: IUSPIM 1996

⟦NIPS 1⟧ D. S. Touretzky (Hrsg.): *Advances in Neural Information Processing Systems 1*. San Mateo: Morgan Kaufmann Publishers 1989

⟦NIPS 4⟧ J. E. Moody, S. J. Hanson und R. P. Lippmann (Hrsg.): *Advances in Neural Information Processing Systems 4*. San Mateo: Morgan Kaufmann Publishers 1992

⟦NIPS 5⟧ S. J. Hanson, J. D. Cowan und C. L. Giles (Hrsg.): *Advances in Neural Information Processing Systems 5*. San Mateo: Morgan Kaufmann Publishers 1993

⟦NIPS 6⟧ J. D. Cowan, G. Tesauro und J. Alspector (Hrsg.): *Advances in Neural Information Processing Systems 6*. San Mateo: Morgan Kaufmann Publishers 1994

⟦NIPS 7⟧ G. Tesauro, D. S. Touretzky und T. K. Leen (Hrsg.): *Advances in Neural Information Processing Systems 7*. Cambridge: MIT Press 1995

[[NIPS 8]] D. S. Touretzky, M. C. Mozer und M. E. Hasselmo (Hrsg.): *Advances in Neural Information Processing Systems 8*. Cambridge: MIT Press 1996

[[Onsager 1944]] L. Onsager: *Crystal Statistics. I. A Two-Dimensional Model with an Order-Disorder Transition*. Physical Review 65 (3 und 4), 117–149

[[Ossen 1996]] A. Ossen: private Diskussion

[[Ossen und Rüger 1996b]] A. Ossen und S. M. Rüger: *An Analysis of the Metric Structure of the Weight Space of Feedforward Networks and its Application to Time Series Modeling and Prediction*. In [[ESANN-95]], 315–322

[[Ossen und Rüger 1996d]] A. Ossen und S. M. Rüger: *Weight Space Analysis and Forecast Uncertainty*. Journal of Forecasting (angenommen)

[[Ossen und Rüger 1996f]] A. Ossen und S. M. Rüger: *Robust Model Selection for Neural Networks*. In Vorbereitung

[[Paaß 1994]] G. Paaß: *Prognosegenauigkeit und Modellauswahl bei Neuronalen Netzen*. In [[HeKoNN-94]], 41–59

[[Peterson und Anderson 1987]] C. Peterson und J. R. Anderson: *A Mean Field Theory Learning Algorithm for Neural Networks*. Complex Systems 1, 995–1019

[[Pfanzagl 1991]] J. Pfanzagl: *Elementare Wahrscheinlichkeitsrechnung*. Berlin: de Gruyter

[[Pfeifer et al. 1989]] R. Pfeifer, Z. Schreter, F. Fogelman-Soulié und L. Steels (Hrsg.): *Connectionism in Perspective*. Amsterdam: Elsevier Science Publishers

[[Pfister und Rojas 1993]] M. Pfister und R. Rojas: *Speeding-up Backpropagation — A Comparison of orthogonal Techniques*. In [[IJCNN-93]], 517-523

[[Pfister und Rojas 1996]] M. Pfister und R. Rojas: *Hybrid Learning Algorithms for Neural Networks – The adaptive Inclusion of Second Order Information*. Zeitschrift für Angewandte Mathematik und Mechanik (noch nicht erschienen)

[[Polak 1971]] E. Polak: *Computational Methods in Optimization. A Unified Approach.* New York: Academic Press

[[Polak und Ribière 1969]] E. Polak und G. Ribière: *Note sur la convergence de méthodes de directions conjugées*. Rev. Fr. Inform. Rech. Operation (16-R1), 35–43

[[Press et al. 1988]] W. H. Press, S. A. Teukolsky, W. T. Vetterling und B. P. Flannery: *Numerical Recipes in C.* Cambridge University Press

[[Radons et al. 1990]] G. Radons, H. G. Schuster und D. Werner: *Drift and Diffusion in Backpropagation Networks.* In [[ICNC-90]], 261–264

[[Resnikoff 1989]] H. L. Resnikoff: *The Illusion of Reality.* New York: Springer-Verlag

[[Ripley 1996]] B. D. Ripley: *Pattern Recognition and Neural Networks.* Cambridge University Press

[[Ritter et al. 1990]] H. Ritter, T. Martinetz und K. Schulten: *Eine Einführung in die Neuroinformatik selbstorganisierender Netzwerke.* Redwood City: Addison-Wesley

[[Rojas 1993]] R. Rojas: *Theorie der neuronalen Netze. Eine systematische Einführung.* Berlin: Springer-Verlag

[[Rüger 1996a]] S. M. Rüger: *Ausgewählte Kapitel aus der Theorie Neuronaler Netze.* Technische Universität Berlin: Vorlesungsskript

[[Rüger 1996b]] S. M. Rüger: *Stable Dynamic Parameter Adaptation.* In [[NIPS 8]], 225–231

[[Rüger 1996c]] S. M. Rüger: *Effizientes Lernen und Schließen in dezimierbaren Boltzmann-Maschinen.* In [[CoWAN-96]] (noch nicht erschienen)

[[Rüger et al. 1996]] S. M. Rüger, A. Weinberger und S. Wittchen: *Decimatable Boltzmann Machines vs. Gibbs Sampling.* Technische Universität Berlin: Bericht 96-29 des Fachbereichs Informatik

[[Rüger und Ossen 1996a]] S. M. Rüger und A. Ossen: *Performance Evaluation of Feedforward Networks Using Computational Methods.* In [[NEURAP-95]], 35–39

[[Rüger und Ossen 1996c]] S. M. Rüger und A. Ossen: *Clustering in Weight Space of Feedforward Nets.* In [[ICANN-96]], 83–88

[[Rüger und Ossen 1996e]] S. M. Rüger und A. Ossen: *The Metric Structure of Weight Space*. In Vorbereitung

[[Rumelhart et al. 1986a]] D. E. Rumelhart, G. E. Hinton und R. J. Williams: *Learning Internal Representations by Error Propagation*. In [[Rumelhart et al. 1986b]], 318–364

[[Rumelhart et al. 1986b]] D. E. Rumelhart, J. L. McClelland und die PDP Reasearch Group (Hrsg.): *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1: Foundations*. Cambridge: MIT Press

[[Salomon 1991]] R. Salomon: *Verbesserung konnektionistischer Lernverfahren, die nach der Gradientenmethode arbeiten*. Technische Universität Berlin: Dissertation

[[Salomon und van Hemmen 1996]] R. Salomon und J. L. van Hemmen: *Accelerating Backpropagation through Dynamic Self-Adaptation*. Neural Networks 9 (4), 589–601

[[Sarle 1994]] W. S. Sarle: *Neural Networks and Statistical Models*. In Proceedings of the Nineteenth Annual SAS Users Group International Conference, 1538–1550, Cary: SAS Institute

[[Saul und Jordan 1994]] L. K. Saul und M. I. Jordan: *Learning in Boltzmann Trees*. Neural Computation 6 (6), 1173–1183

[[Saul und Jordan 1995]] L. K. Saul und M. I. Jordan: *Boltzmann Chains and Hidden Markov Models*. In [[NIPS 7]], 435–442

[[Sussmann 1992]] H. J. Sussmann: *Uniqueness of the Weights for Minimal Feedforward Nets with a Given Input-Output Map*. Neural Networks 5, 589–593

[[Syozi 1972]] I. Syozi: *Transformation of Ising Models*. In [[Domb und Green 1972]], 269–329

[[Szpilrajn 1930]] E. Szpilrajn: Fundamenta Mathematica 16, 386–389

[[Taylor 1993]] J. G. Taylor (Hrsg.): *„Mathematical Approaches to Neural Networks“*. Amsterdam: Elsevier Science Publishers

[[Tesauro 1989]] G. Tesauro: *Connectionist Learning of Expert Preferences by Comparison Training*. In [[NIPS 1]], 99–106

[[Tibshirani 1995]] R. J. Tibshirani: private Mitteilung

[[Tibshirani 1996]] R. J. Tibshirani: *A Comparison of Some Error Estimates for Neural Network Models*. Neural Computation 8 (1), 152–163

[[van Hemmen et al. 1990]] J. L. van Hemmen, L. B. Ioffe, R. Kühn und M. Vaas: *Increasing the Efficiency of a Neural Network through Unlearning*. Physica 163A, 386–392

[[White 1989a]] H. White: *Neural Network Learning and Statistics*. AI Expert (Dez.), 48–52

[[White 1989b]] H. White: *Learning in Artificial Neural Networks: a Statistical Perspective*. Journal of the American Statistical Association 84, 1008–1013

[[White 1992]] H. White: *Artificial Neural Networks. Approximation & Learning Theory*. Oxford: Blackwell

[[Widrow und Lehr 1992]] B. Widrow und M. A. Lehr: *30 Years of Adaptive Neural Networks: Perceptron, Madaline, and Backpropagation*. In [[Lau 1992]], 27–53

[[Williams und Zipser 1989]] R. J. Williams und D. Zipser: *A Learning Algorithm for Continually Running Fully Recurrent Neural Networks*. Neural Computation 1, 270–280

Symbolverzeichnis

$\overline{W}$	Abschluß einer Menge	4.3.5
$\overline{X}^r$	Mittelwert von X bezüglich r	5.2.4
(a, b)	Kante	1.1.1
a, b, c	Knoten	1.1.1
α	Impulsparameter	2.1.7
α	Zustand sichtbarer Knoten	5.2.1
α	Zustand der Eingabeknoten	5.3.3
B	Anzahl von Bootstrap-Stichproben	3.4.1
$b(G, w)$	Bias eines Schätzers	3.2.1
β	Zustand von Binnenknoten	5.2.1
C_m^r	Kumulante	6.2.3
$\mathrm{C}(S_1^{m_1} \dots S_n^{m_n})$	meint die Kumulante C_m^r	6.2.3
D	Jacobimatrix	nach 1.1.7
$\mathrm{D}^{(i)}$	i-te Ableitung	2.1.1
d_a	Ableitung der Aktivierungsfunktion f_a	1.3.1
d_E	Metrik in $\mathbb{R}^E$	4.4.3
$d(v, w)$	kanonische Metrik im Fundamentalgebiet	4.4.4
$d(x, y)$	Abweichung von Vektoren, Kostenfunktion	1.2.8
ΔH_c	Flip-Energie	5.1.6
Δw_{ab}	Gewichtungsänderung durch eine Lernregel	5.2.4
δ_{ij}	Kronecker-Symbol	1.6.2
E	Menge der Kanten eines Graphen (Netzes)	1.1.1
E^α	Kantenmenge von $(N^\alpha, E^\alpha, \dots)$	vor 6.3.1
E^*	vollverknüpfte Kantenmenge	6.6.5
$E(\bullet)$	n-dimensionale Fehlerfunktion	2.1.2
$E^i(\bullet)$	i-ter Einzelfehler	2.6.3
$E_{\min}$	unteres Limit für den Fehler	2.3.3
E_w	relative Entropie	5.2.1

$\mathrm{E}(\bullet)$	Erwartungswert bez. Boltzmann-Gibbs-Vert.	5.1.7
$\mathrm{E}(\bullet)_\alpha$	Erwartungswert bez. Boltzmann-Gibbs-Vert. gegeben die festen Aktivationen α	5.2.4
$\mathrm{E}_w(\bullet)$	Erwartungswert bez. Dichte $T \mapsto L_T(w)$	3.2.1
ϵ^i	unabhängige Zufallsvariablen (Meßfehler)	3.1.2
Err_T	Gesamtfehler des Netzes (abhängig von w)	1.2.8
$\mathrm{err}_w(p,t)$	Fehler des Netzes an einem Testmuster	1.2.8
η	Lernrate	1.2.9
f_a	Aktivierungsfunktion des Knotens a	1.2.1
G	Testmultimenge	4.6.5
g	(zu schätzende) Funktion eines Parameters	3.1.3
g^t	Vektor aus dem Gradientenkegel	2.3.3
γ	Konfidenzniveau	3.2.2
γ	Zustand der Ausgabeknoten	5.3.3
$\mathrm{GL}(E,\mathbb{R})$	Gruppe linearer Abbildungen $\mathbb{R}^E \to \mathbb{R}^E$	vor 4.2.1
H	Hessesche Matrix	2.1.1
h_a	Eingabevektor eines Knotens $a \in N \setminus I$	1.1.6
H^α	Energie von $(N^\alpha, E^\alpha, \ldots)$	vor 6.3.1
$H(w,s)$	Energie einer Boltzmann-Maschine	5.1.5
I	Menge der Eingabeknoten	1.1.5
i	imaginäre Einheit	6.2.1
I	Einheitsmatrix	2.1.1
I_w	Information gain	5.3.4
k	Anzahl der Binnenschichten	1.1.1
k	Boltzmann-Konstante $1{,}380662 \cdot 10^{-23}$ J/K	Vorwort
L	Länge der Hidden-Markow-Kette	6.4.6
L_a	Menge der Vorgängerknoten	1.1.5
$L_T(w)$	Likelihood eines Vektors w	3.1.3
λ_e	e-ter Eigenwert	2.1.1
m	Anzahl der Versuchsläufe eines Experiments	4.5.1
m_a	Zahl der mögl. Aktivationen des Knotens a	6.4.1
N	Menge der Knoten eines Graphen (Netzes)	1.1.1
n	Dimension (Fehlerfunktion $E\colon \mathbb{R}^n \to \mathbb{R}$)	2.1.2

N_*	Menge der Knoten ohne Biasknoten	5.1.7
N^α	Knotenmenge von $(N^\alpha, E^\alpha, \ldots)$	vor 6.3.1
N^α	symbolische Abkürzung für $(N^\alpha, E^\alpha, \ldots)$	vor 6.3.1
(N, E)	Graph eines neuronalen Netzes	1.1.1
(N, E, f, w, O)	neuronales Abbildungsnetz	1.2.1
(N, E, f, w, O, d)	Backpropagation-Netz	1.2.8
(N, E, o, O)	Abbildungsnetz	1.1.6
(N, E, w, T)	Boltzmann-Maschine	5.1.1
$(N^\alpha, E^\alpha, \ldots)$	Boltzmann-Maschine mit aufgeprägtem α	vor 6.3.1
(N, E, X, P_X)	Belief-Netz	6.6.1
(N, w)	Hopfield-Netz	5.3.5
$\mathrm{nearest}(v, w)$	zu w nächstes Element aus $S(v)$	4.4.10
O	Menge der Ausgabeknoten	1.1.6
o_a	Aktivation eines Knotens a	1.1.6
out	Schar von Netzfunktionen	1.1.6
$\overline{\mathrm{out}}^B$	mittlere Bootstrap-Regressionslinie	3.4.1
p	Eingabevektor eines Netzes	1.1.6
p	Dichte	6.2.1
p_a^i	Mustersequenz am Knoten a	vor 1.6.1
P_S	Zu S gehöriges Wahrscheinlichkeitsmaß	6.2.1
(p, t)	Eingabe- und Ausgabemuster	1.2.8
p_w	Randverteilung an den sichtbaren Knoten	5.2.1
P_w	Wahrscheinlichkeitsmaß zur Dichte $L_\bullet(w)$	3.2.2
P_w	Boltzmann-Gibbs-Verteilung	5.1.9
$P^\alpha_{w^\alpha}(\beta\gamma)$	$P_w(\beta\gamma\|\alpha)$	vor 6.3.1
$P_w(\beta\|\alpha)$	bedingte Wahrscheinlichkeit	5.2.4
Φ	Phasenraumvolumen	Vorwort
ϕ^α_Γ	charakteristische Funktion für $(N^\alpha, E^\alpha, \ldots)$	vor 6.3.1
Φ_S	kumulantengenerierende Funktion	6.2.3
ϕ_S	charakteristische Funktion	6.2.1
π	3,1415…	2.3.4
Π	von $\Pi_1, \ldots, \Pi_k$ erzeugte Gruppe	4.2.5
π	Permutationssymmetrie	4.1.2

Π	anfängliche Wahrscheinlichkeiten (HMM)	6.4.6
Π_i	Permutationsgruppe für Binnenschicht i	4.2.1
Π_k	Projektion: k-te Komponente eines Tupels	1.1.5
q	Randverteilung der Eingabemuster	5.3.3
Q	quadratische Form	2.1.1
Q	Matrix von Übergangswahrscheinlichkeiten	5.1.7
r	gewünschte Wahrscheinlichkeitsverteilung	5.2.1
R_a	Menge der Nachfolgerknoten	1.1.5
$R(G, w)$	Risiko eines Schätzers	3.2.1
$r(\hat{w})$	Repräsentantenvektor	4.3.5
P	Übergangswahrscheinlichkeitsmatrix (HMM)	6.4.6
S	Entropie	Vorwort
S	Pattern-Target-Relation	1.2.8
S	Symmetriegruppe	vor 4.2.1
s	Symmetrie	4.1.1
S	Menge der sichtbaren Knoten	5.1.1
S	$\mathbb{R}^n$-wertige Zufallsvariable	6.2.1
$s(a)$	Schichtindex	1.1.9
s_a	Aktivation eines Knotens	5.1.1
$s, s', \ldots$	Zustand einer Boltzmann-Maschine	5.1.1
$S(w)$	Orbit von $w \in \mathbb{R}^E$	4.3.1
Σ	Varianz-Kovarianzmatrix	3.3.3
Σ	Ausgabewahrscheinlichkeitsmatrix (HMM)	6.4.6
σ	Fermifunktion	vor 1.5.1
$\Sigma(M)$	Permutationsgruppe einer Menge M	4.2.1
T	Menge von Trainingsmustern (mit Target)	1.2.8
T	von allen T_b erzeugte Gruppe	4.2.5
T_a	Spiegelsymmetrie-Gruppe am Knoten a	4.2.2
t_a	Spiegelsymmetrie am Knoten a	4.1.2
T^{*i}	Bootstrap-Stichprobe	3.4.1
θ	Sprungfunktion	5.1.3
U	Menge der unsichtbaren Knoten	5.1.1
v	effektiver Gewichtungssatz w/T	6.1.1

W	Fundamentalgebiet	4.3.2
w	Gewichtungssatz, w_{ab} ist ein Gewicht	1.2.1
$w_{ab}(s_a, s_b)$	Gewichtsmatrixelement	6.4.1
w^α	Gewichtungssatz von $(N^\alpha, E^\alpha, \ldots)$	vor 6.3.1
$\hat{w}^{*i}$	Bootstrap-Schätzung	3.4.1
$\hat{w}_n$	Maximum-Likelihood-Schätzer	3.1.3
$w^\star$	Gewichtungsvektor des „wahren“ Modells	3.1.2
w^*	Optimum im Gewichtungsraum	2.1.1
$\tilde{w}(a, i)$	Teilvektor von w am Knoten a	4.1.2
y	Ausgabevektor eines Netzes	1.1.6
Z_w	Zustandssumme	5.1.9
$Z^\alpha_{w^\alpha}$	Zustandssumme von $(N^\alpha, E^\alpha, \ldots)$	vor 6.3.1

Namen- und Sachverzeichnis

Catherine: *Je m'en fais la répétition de tous les mots, que vous m'avez appris dès à présent.*

— Shakespeare, Heinrich V (III. Akt, 4. Szene)

C

D

E

F

G

H

I

J

K

L

M

N

S

Z

Nachwort

On ne finit pas un œuvre,
on l'abandonne.
— G. Flaubert nach [[Bauer 1991, S. 500]]

An dieser Stelle sage ich all jenen Dank, die zum Gelingen dieser Arbeit beigetragen haben: *Prof. Biedl* war stets bereit, sich mit mir über die Darstellung und die Inhalte auseinanderzusetzen; seine wertvollen Hinweise habe ich gerne aufgenommen. Die Arbeit verdankt ihren Umfang dem intensiven Bestreben von *Prof. Obermayer* danach, daß ich auch jene Themen in die Dissertation aufnehme, die ich bereits vorher zum Gebiet der neuronalen Netze veröffentlicht habe. Dadurch wurde ich angespornt, ein weit größeres Gebiet abzudecken als es ursprünglich meine Intention war. Ebenfalls möchte ich mich bei *Prof. Kockelkorn* bedanken, der von Anfang an das Entstehen dieser Arbeit mit intensivem Interesse begleitet hat. Nicht zuletzt gilt mein Dank *Dr. Ossen*, vor allem für die angenehme und gewinnbringende Zusammenarbeit. *smr*